



Intel® Xeon Phi™ Processor

Performance Monitoring Reference Manual – Volume 1: Registers

Rev. 1.0

June 2016



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit <http://www.intel.com/design/literature.htm>.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at <http://www.intel.com/> or from the OEM or retailer.

No computer system can be absolutely secure.

Intel, the Intel logo, Intel Core, Intel SpeedStep, Intel Xeon Phi, Pentium, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2016, Intel Corporation. All rights reserved.



Revision History

Document Number	Revision Number	Description	Revision Date
332972	1.0	Updates to Chapter 1	June 2016
332972	0.6	Initial Release	November 2015



Contents

Revision History.....	3
1.0 Performance Monitoring in the Intel® Xeon Phi™ processor - Introduction.....	26
1.1 Intel® Xeon Phi™ Processor Architecture Overview.....	26
1.1.1 Intel® Xeon Phi™ processor Memory and Cluster Modes.....	28
1.2 Performance Monitoring in the Intel® Xeon Phi™ processor tile.....	29
1.2.1 Enhancements of Performance Monitoring in the Intel® Xeon Phi™ processor Tile.....	29
1.3 Performance Monitoring in the Intel® Xeon Phi™ processor Untile.....	32
1.3.1 Untile Performance Monitoring Registers.....	33
1.4 Terminology.....	33
1.5 Register Terminology.....	36
2.0 Embedded DRAM Controller (EDC) Registers.....	39
2.1 Bus: 2, Device: 15, Function: 0 (CFG).....	40
2.1.1 UCLK_PMON_CTR0_LOW_REG.....	41
2.1.2 UCLK_PMON_CTR0_HIGH_REG.....	41
2.1.3 UCLK_PMON_CTR1_LOW_REG.....	41
2.1.4 UCLK_PMON_CTR1_HIGH_REG.....	41
2.1.5 UCLK_PMON_CTR2_LOW_REG.....	42
2.1.6 UCLK_PMON_CTR2_HIGH_REG.....	42
2.1.7 UCLK_PMON_CTR3_LOW_REG.....	42
2.1.8 UCLK_PMON_CTR3_HIGH_REG.....	42
2.1.9 UCLK_PMON_CTRCTLO_REG.....	42
2.1.10 UCLK_PMON_CTRCTL1_REG.....	43
2.1.11 UCLK_PMON_CTRCTL2_REG.....	44
2.1.12 UCLK_PMON_CTRCTL3_REG.....	45
2.1.13 UCLK_PMON_UNIT_CTL_REG.....	46
2.1.14 UCLK_PMON_UNIT_STATUS_REG.....	47
2.1.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	47
2.1.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	47
2.1.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	48
2.2 Bus: 2, Device: 16, Function: 0 (CFG).....	48
2.2.1 UCLK_PMON_CTR0_LOW_REG.....	49
2.2.2 UCLK_PMON_CTR0_HIGH_REG.....	49
2.2.3 UCLK_PMON_CTR1_LOW_REG.....	49
2.2.4 UCLK_PMON_CTR1_HIGH_REG.....	49
2.2.5 UCLK_PMON_CTR2_LOW_REG.....	49
2.2.6 UCLK_PMON_CTR2_HIGH_REG.....	50
2.2.7 UCLK_PMON_CTR3_LOW_REG.....	50
2.2.8 UCLK_PMON_CTR3_HIGH_REG.....	50
2.2.9 UCLK_PMON_CTRCTLO_REG.....	50
2.2.10 UCLK_PMON_CTRCTL1_REG.....	51
2.2.11 UCLK_PMON_CTRCTL2_REG.....	52
2.2.12 UCLK_PMON_CTRCTL3_REG.....	53
2.2.13 UCLK_PMON_UNIT_CTL_REG.....	54
2.2.14 UCLK_PMON_UNIT_STATUS_REG.....	54
2.2.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	55



2.2.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	55
2.2.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	55
2.3 Bus: 2, Device: 17, Function: 0 (CFG).....	55
2.3.1 UCLK_PMON_CTR0_LOW_REG.....	56
2.3.2 UCLK_PMON_CTR0_HIGH_REG.....	56
2.3.3 UCLK_PMON_CTR1_LOW_REG.....	56
2.3.4 UCLK_PMON_CTR1_HIGH_REG.....	56
2.3.5 UCLK_PMON_CTR2_LOW_REG.....	57
2.3.6 UCLK_PMON_CTR2_HIGH_REG.....	57
2.3.7 UCLK_PMON_CTR3_LOW_REG.....	57
2.3.8 UCLK_PMON_CTR3_HIGH_REG.....	57
2.3.9 UCLK_PMON_CTRCTL0_REG.....	57
2.3.10 UCLK_PMON_CTRCTL1_REG.....	58
2.3.11 UCLK_PMON_CTRCTL2_REG.....	59
2.3.12 UCLK_PMON_CTRCTL3_REG.....	60
2.3.13 UCLK_PMON_UNIT_CTL_REG.....	61
2.3.14 UCLK_PMON_UNIT_STATUS_REG.....	62
2.3.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	62
2.3.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	62
2.3.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	63
2.4 Bus: 2, Device: 18, Function: 0 (CFG).....	63
2.4.1 UCLK_PMON_CTR0_LOW_REG.....	64
2.4.2 UCLK_PMON_CTR0_HIGH_REG.....	64
2.4.3 UCLK_PMON_CTR1_LOW_REG.....	64
2.4.4 UCLK_PMON_CTR1_HIGH_REG.....	64
2.4.5 UCLK_PMON_CTR2_LOW_REG.....	64
2.4.6 UCLK_PMON_CTR2_HIGH_REG.....	65
2.4.7 UCLK_PMON_CTR3_LOW_REG.....	65
2.4.8 UCLK_PMON_CTR3_HIGH_REG.....	65
2.4.9 UCLK_PMON_CTRCTL0_REG.....	65
2.4.10 UCLK_PMON_CTRCTL1_REG.....	66
2.4.11 UCLK_PMON_CTRCTL2_REG.....	67
2.4.12 UCLK_PMON_CTRCTL3_REG.....	68
2.4.13 UCLK_PMON_UNIT_CTL_REG.....	69
2.4.14 UCLK_PMON_UNIT_STATUS_REG.....	69
2.4.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	70
2.4.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	70
2.4.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	70
2.5 Bus: 2, Device: 19, Function: 0 (CFG).....	70
2.5.1 UCLK_PMON_CTR0_LOW_REG.....	71
2.5.2 UCLK_PMON_CTR0_HIGH_REG.....	71
2.5.3 UCLK_PMON_CTR1_LOW_REG.....	71
2.5.4 UCLK_PMON_CTR1_HIGH_REG.....	71
2.5.5 UCLK_PMON_CTR2_LOW_REG.....	72
2.5.6 UCLK_PMON_CTR2_HIGH_REG.....	72
2.5.7 UCLK_PMON_CTR3_LOW_REG.....	72
2.5.8 UCLK_PMON_CTR3_HIGH_REG.....	72
2.5.9 UCLK_PMON_CTRCTL0_REG.....	72
2.5.10 UCLK_PMON_CTRCTL1_REG.....	73
2.5.11 UCLK_PMON_CTRCTL2_REG.....	74
2.5.12 UCLK_PMON_CTRCTL3_REG.....	75



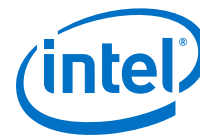
2.5.13 UCLK_PMON_UNIT_CTL_REG.....	76
2.5.14 UCLK_PMON_UNIT_STATUS_REG.....	77
2.5.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	77
2.5.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	77
2.5.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	78
2.6 Bus: 2, Device: 20, Function: 0 (CFG).....	78
2.6.1 UCLK_PMON_CTR0_LOW_REG.....	79
2.6.2 UCLK_PMON_CTR0_HIGH_REG.....	79
2.6.3 UCLK_PMON_CTR1_LOW_REG.....	79
2.6.4 UCLK_PMON_CTR1_HIGH_REG.....	79
2.6.5 UCLK_PMON_CTR2_LOW_REG.....	79
2.6.6 UCLK_PMON_CTR2_HIGH_REG.....	80
2.6.7 UCLK_PMON_CTR3_LOW_REG.....	80
2.6.8 UCLK_PMON_CTR3_HIGH_REG.....	80
2.6.9 UCLK_PMON_CTRCTL0_REG.....	80
2.6.10 UCLK_PMON_CTRCTL1_REG.....	81
2.6.11 UCLK_PMON_CTRCTL2_REG.....	82
2.6.12 UCLK_PMON_CTRCTL3_REG.....	83
2.6.13 UCLK_PMON_UNIT_CTL_REG.....	84
2.6.14 UCLK_PMON_UNIT_STATUS_REG.....	84
2.6.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	85
2.6.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	85
2.6.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	85
2.7 Bus: 2, Device: 21, Function: 0 (CFG).....	85
2.7.1 UCLK_PMON_CTR0_LOW_REG.....	86
2.7.2 UCLK_PMON_CTR0_HIGH_REG.....	86
2.7.3 UCLK_PMON_CTR1_LOW_REG.....	86
2.7.4 UCLK_PMON_CTR1_HIGH_REG.....	86
2.7.5 UCLK_PMON_CTR2_LOW_REG.....	87
2.7.6 UCLK_PMON_CTR2_HIGH_REG.....	87
2.7.7 UCLK_PMON_CTR3_LOW_REG.....	87
2.7.8 UCLK_PMON_CTR3_HIGH_REG.....	87
2.7.9 UCLK_PMON_CTRCTL0_REG.....	87
2.7.10 UCLK_PMON_CTRCTL1_REG.....	88
2.7.11 UCLK_PMON_CTRCTL2_REG.....	89
2.7.12 UCLK_PMON_CTRCTL3_REG.....	90
2.7.13 UCLK_PMON_UNIT_CTL_REG.....	91
2.7.14 UCLK_PMON_UNIT_STATUS_REG.....	92
2.7.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	92
2.7.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	92
2.7.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	93
2.8 Bus: 2, Device: 22, Function: 0 (CFG).....	93
2.8.1 UCLK_PMON_CTR0_LOW_REG.....	94
2.8.2 UCLK_PMON_CTR0_HIGH_REG.....	94
2.8.3 UCLK_PMON_CTR1_LOW_REG.....	94
2.8.4 UCLK_PMON_CTR1_HIGH_REG.....	94
2.8.5 UCLK_PMON_CTR2_LOW_REG.....	94
2.8.6 UCLK_PMON_CTR2_HIGH_REG.....	95
2.8.7 UCLK_PMON_CTR3_LOW_REG.....	95
2.8.8 UCLK_PMON_CTR3_HIGH_REG.....	95
2.8.9 UCLK_PMON_CTRCTL0_REG.....	95



2.8.10 UCLK_PMON_CTLCTL1_REG.....	96
2.8.11 UCLK_PMON_CTLCTL2_REG.....	97
2.8.12 UCLK_PMON_CTLCTL3_REG.....	98
2.8.13 UCLK_PMON_UNIT_CTL_REG.....	99
2.8.14 UCLK_PMON_UNIT_STATUS_REG.....	99
2.8.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	100
2.8.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	100
2.8.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	100
2.9 Bus: 2, Device: 24, Function: 2 (CFG).....	100
2.9.1 ECLK_PMON_CTR0_LOW_REG.....	101
2.9.2 ECLK_PMON_CTR0_HIGH_REG.....	101
2.9.3 ECLK_PMON_CTR1_LOW_REG.....	101
2.9.4 ECLK_PMON_CTR1_HIGH_REG.....	101
2.9.5 ECLK_PMON_CTR2_LOW_REG.....	102
2.9.6 ECLK_PMON_CTR2_HIGH_REG.....	102
2.9.7 ECLK_PMON_CTR3_LOW_REG.....	102
2.9.8 ECLK_PMON_CTR3_HIGH_REG.....	102
2.9.9 ECLK_PMON_CTLCTL0_REG.....	102
2.9.10 ECLK_PMON_CTLCTL1_REG.....	103
2.9.11 ECLK_PMON_CTLCTL2_REG.....	104
2.9.12 ECLK_PMON_CTLCTL3_REG.....	105
2.9.13 ECLK_PMON_UNIT_CTL_REG.....	106
2.9.14 ECLK_PMON_UNIT_STATUS_REG.....	107
2.9.15 ECLK_PMON_TIMESTAMP_LOW_REG.....	107
2.9.16 ECLK_PMON_TIMESTAMP_HIGH_REG.....	107
2.9.17 ECLK_PMON_TIMESTAMP_CTL_REG.....	108
2.10 Bus: 2, Device: 25, Function: 2 (CFG).....	108
2.10.1 ECLK_PMON_CTR0_LOW_REG.....	109
2.10.2 ECLK_PMON_CTR0_HIGH_REG.....	109
2.10.3 ECLK_PMON_CTR1_LOW_REG.....	109
2.10.4 ECLK_PMON_CTR1_HIGH_REG.....	109
2.10.5 ECLK_PMON_CTR2_LOW_REG.....	109
2.10.6 ECLK_PMON_CTR2_HIGH_REG.....	110
2.10.7 ECLK_PMON_CTR3_LOW_REG.....	110
2.10.8 ECLK_PMON_CTR3_HIGH_REG.....	110
2.10.9 ECLK_PMON_CTLCTL0_REG.....	110
2.10.10 ECLK_PMON_CTLCTL1_REG.....	111
2.10.11 ECLK_PMON_CTLCTL2_REG.....	112
2.10.12 ECLK_PMON_CTLCTL3_REG.....	113
2.10.13 ECLK_PMON_UNIT_CTL_REG.....	114
2.10.14 ECLK_PMON_UNIT_STATUS_REG.....	114
2.10.15 ECLK_PMON_TIMESTAMP_LOW_REG.....	115
2.10.16 ECLK_PMON_TIMESTAMP_HIGH_REG.....	115
2.10.17 ECLK_PMON_TIMESTAMP_CTL_REG.....	115
2.11 Bus: 2, Device: 26, Function: 2 (CFG).....	115
2.11.1 ECLK_PMON_CTR0_LOW_REG.....	116
2.11.2 ECLK_PMON_CTR0_HIGH_REG.....	116
2.11.3 ECLK_PMON_CTR1_LOW_REG.....	116
2.11.4 ECLK_PMON_CTR1_HIGH_REG.....	116
2.11.5 ECLK_PMON_CTR2_LOW_REG.....	117
2.11.6 ECLK_PMON_CTR2_HIGH_REG.....	117



2.11.7	ECLK_PMON_CTR3_LOW_REG.....	117
2.11.8	ECLK_PMON_CTR3_HIGH_REG.....	117
2.11.9	ECLK_PMON_CTRCTL0_REG.....	117
2.11.10	ECLK_PMON_CTRCTL1_REG.....	118
2.11.11	ECLK_PMON_CTRCTL2_REG.....	119
2.11.12	ECLK_PMON_CTRCTL3_REG.....	120
2.11.13	ECLK_PMON_UNIT_CTL_REG.....	121
2.11.14	ECLK_PMON_UNIT_STATUS_REG.....	122
2.11.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	122
2.11.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	122
2.11.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	123
2.12	Bus: 2, Device: 27, Function: 2 (CFG).....	123
2.12.1	ECLK_PMON_CTR0_LOW_REG.....	124
2.12.2	ECLK_PMON_CTR0_HIGH_REG.....	124
2.12.3	ECLK_PMON_CTR1_LOW_REG.....	124
2.12.4	ECLK_PMON_CTR1_HIGH_REG.....	124
2.12.5	ECLK_PMON_CTR2_LOW_REG.....	124
2.12.6	ECLK_PMON_CTR2_HIGH_REG.....	125
2.12.7	ECLK_PMON_CTR3_LOW_REG.....	125
2.12.8	ECLK_PMON_CTR3_HIGH_REG.....	125
2.12.9	ECLK_PMON_CTRCTL0_REG.....	125
2.12.10	ECLK_PMON_CTRCTL1_REG.....	126
2.12.11	ECLK_PMON_CTRCTL2_REG.....	127
2.12.12	ECLK_PMON_CTRCTL3_REG.....	128
2.12.13	ECLK_PMON_UNIT_CTL_REG.....	129
2.12.14	ECLK_PMON_UNIT_STATUS_REG.....	129
2.12.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	130
2.12.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	130
2.12.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	130
2.13	Bus: 2, Device: 28, Function: 2 (CFG).....	130
2.13.1	ECLK_PMON_CTR0_LOW_REG.....	131
2.13.2	ECLK_PMON_CTR0_HIGH_REG.....	131
2.13.3	ECLK_PMON_CTR1_LOW_REG.....	131
2.13.4	ECLK_PMON_CTR1_HIGH_REG.....	131
2.13.5	ECLK_PMON_CTR2_LOW_REG.....	132
2.13.6	ECLK_PMON_CTR2_HIGH_REG.....	132
2.13.7	ECLK_PMON_CTR3_LOW_REG.....	132
2.13.8	ECLK_PMON_CTR3_HIGH_REG.....	132
2.13.9	ECLK_PMON_CTRCTL0_REG.....	132
2.13.10	ECLK_PMON_CTRCTL1_REG.....	133
2.13.11	ECLK_PMON_CTRCTL2_REG.....	134
2.13.12	ECLK_PMON_CTRCTL3_REG.....	135
2.13.13	ECLK_PMON_UNIT_CTL_REG.....	136
2.13.14	ECLK_PMON_UNIT_STATUS_REG.....	137
2.13.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	137
2.13.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	137
2.13.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	138
2.14	Bus: 2, Device: 29, Function: 2 (CFG).....	138
2.14.1	ECLK_PMON_CTR0_LOW_REG.....	139
2.14.2	ECLK_PMON_CTR0_HIGH_REG.....	139
2.14.3	ECLK_PMON_CTR1_LOW_REG.....	139



2.14.4	ECLK_PMON_CTR1_HIGH_REG.....	139
2.14.5	ECLK_PMON_CTR2_LOW_REG.....	139
2.14.6	ECLK_PMON_CTR2_HIGH_REG.....	140
2.14.7	ECLK_PMON_CTR3_LOW_REG.....	140
2.14.8	ECLK_PMON_CTR3_HIGH_REG.....	140
2.14.9	ECLK_PMON_CTRCTL0_REG.....	140
2.14.10	ECLK_PMON_CTRCTL1_REG.....	141
2.14.11	ECLK_PMON_CTRCTL2_REG.....	142
2.14.12	ECLK_PMON_CTRCTL3_REG.....	143
2.14.13	ECLK_PMON_UNIT_CTL_REG.....	144
2.14.14	ECLK_PMON_UNIT_STATUS_REG.....	144
2.14.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	145
2.14.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	145
2.14.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	145
2.15	Bus: 2, Device: 30, Function: 2 (CFG).....	145
2.15.1	ECLK_PMON_CTR0_LOW_REG.....	146
2.15.2	ECLK_PMON_CTR0_HIGH_REG.....	146
2.15.3	ECLK_PMON_CTR1_LOW_REG.....	146
2.15.4	ECLK_PMON_CTR1_HIGH_REG.....	146
2.15.5	ECLK_PMON_CTR2_LOW_REG.....	147
2.15.6	ECLK_PMON_CTR2_HIGH_REG.....	147
2.15.7	ECLK_PMON_CTR3_LOW_REG.....	147
2.15.8	ECLK_PMON_CTR3_HIGH_REG.....	147
2.15.9	ECLK_PMON_CTRCTL0_REG.....	147
2.15.10	ECLK_PMON_CTRCTL1_REG.....	148
2.15.11	ECLK_PMON_CTRCTL2_REG.....	149
2.15.12	ECLK_PMON_CTRCTL3_REG.....	150
2.15.13	ECLK_PMON_UNIT_CTL_REG.....	151
2.15.14	ECLK_PMON_UNIT_STATUS_REG.....	152
2.15.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	152
2.15.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	152
2.15.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	153
2.16	Bus: 2, Device: 31, Function: 2 (CFG).....	153
2.16.1	ECLK_PMON_CTR0_LOW_REG.....	154
2.16.2	ECLK_PMON_CTR0_HIGH_REG.....	154
2.16.3	ECLK_PMON_CTR1_LOW_REG.....	154
2.16.4	ECLK_PMON_CTR1_HIGH_REG.....	154
2.16.5	ECLK_PMON_CTR2_LOW_REG.....	154
2.16.6	ECLK_PMON_CTR2_HIGH_REG.....	155
2.16.7	ECLK_PMON_CTR3_LOW_REG.....	155
2.16.8	ECLK_PMON_CTR3_HIGH_REG.....	155
2.16.9	ECLK_PMON_CTRCTL0_REG.....	155
2.16.10	ECLK_PMON_CTRCTL1_REG.....	156
2.16.11	ECLK_PMON_CTRCTL2_REG.....	157
2.16.12	ECLK_PMON_CTRCTL3_REG.....	158
2.16.13	ECLK_PMON_UNIT_CTL_REG.....	159
2.16.14	ECLK_PMON_UNIT_STATUS_REG.....	159
2.16.15	ECLK_PMON_TIMESTAMP_LOW_REG.....	160
2.16.16	ECLK_PMON_TIMESTAMP_HIGH_REG.....	160
2.16.17	ECLK_PMON_TIMESTAMP_CTL_REG.....	160



3.0 Memory Controller (MC) Registers.....	161
3.1 Bus: 2, Device: 10, Function: 0 (CFG).....	162
3.1.1 UCLK_PMON_CTR0_LOW_REG.....	163
3.1.2 UCLK_PMON_CTR0_HIGH_REG.....	163
3.1.3 UCLK_PMON_CTR1_LOW_REG.....	163
3.1.4 UCLK_PMON_CTR1_HIGH_REG.....	163
3.1.5 UCLK_PMON_CTR2_LOW_REG.....	164
3.1.6 UCLK_PMON_CTR2_HIGH_REG.....	164
3.1.7 UCLK_PMON_CTR3_LOW_REG.....	164
3.1.8 UCLK_PMON_CTR3_HIGH_REG.....	164
3.1.9 UCLK_PMON_CTRCTL0_REG.....	164
3.1.10 UCLK_PMON_CTRCTL1_REG.....	165
3.1.11 UCLK_PMON_CTRCTL2_REG.....	166
3.1.12 UCLK_PMON_CTRCTL3_REG.....	167
3.1.13 UCLK_PMON_UNIT_CTL_REG.....	168
3.1.14 UCLK_PMON_UNIT_STATUS_REG.....	169
3.1.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	169
3.1.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	169
3.1.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	170
3.2 Bus: 2, Device: 8, Function: 2 (CFG).....	170
3.2.1 DCLK_PMON_CTR0_LOW_REG.....	171
3.2.2 DCLK_PMON_CTR0_HIGH_REG.....	171
3.2.3 DCLK_PMON_CTR1_LOW_REG.....	171
3.2.4 DCLK_PMON_CTR1_HIGH_REG.....	171
3.2.5 DCLK_PMON_CTR2_LOW_REG.....	171
3.2.6 DCLK_PMON_CTR2_HIGH_REG.....	172
3.2.7 DCLK_PMON_CTR3_LOW_REG.....	172
3.2.8 DCLK_PMON_CTR3_HIGH_REG.....	172
3.2.9 DCLK_PMON_CTRCTL0_REG.....	172
3.2.10 DCLK_PMON_CTRCTL1_REG.....	173
3.2.11 DCLK_PMON_CTRCTL2_REG.....	174
3.2.12 DCLK_PMON_CTRCTL3_REG.....	175
3.2.13 DCLK_PMON_UNIT_CTL_REG.....	176
3.2.14 DCLK_PMON_UNIT_STATUS_REG.....	176
3.2.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	177
3.2.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	177
3.2.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	177
3.3 Bus: 2, Device: 8, Function: 3 (CFG).....	177
3.3.1 DCLK_PMON_CTR0_LOW_REG.....	178
3.3.2 DCLK_PMON_CTR0_HIGH_REG.....	178
3.3.3 DCLK_PMON_CTR1_LOW_REG.....	178
3.3.4 DCLK_PMON_CTR1_HIGH_REG.....	178
3.3.5 DCLK_PMON_CTR2_LOW_REG.....	179
3.3.6 DCLK_PMON_CTR2_HIGH_REG.....	179
3.3.7 DCLK_PMON_CTR3_LOW_REG.....	179
3.3.8 DCLK_PMON_CTR3_HIGH_REG.....	179
3.3.9 DCLK_PMON_CTRCTL0_REG.....	179
3.3.10 DCLK_PMON_CTRCTL1_REG.....	180
3.3.11 DCLK_PMON_CTRCTL2_REG.....	181
3.3.12 DCLK_PMON_CTRCTL3_REG.....	182



3.3.13 DCLK_PMON_UNIT_CTL_REG.....	183
3.3.14 DCLK_PMON_UNIT_STATUS_REG.....	184
3.3.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	184
3.3.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	184
3.3.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	185
3.4 Bus: 2, Device: 8, Function: 4 (CFG).....	185
3.4.1 DCLK_PMON_CTR0_LOW_REG.....	186
3.4.2 DCLK_PMON_CTR0_HIGH_REG.....	186
3.4.3 DCLK_PMON_CTR1_LOW_REG.....	186
3.4.4 DCLK_PMON_CTR1_HIGH_REG.....	186
3.4.5 DCLK_PMON_CTR2_LOW_REG.....	186
3.4.6 DCLK_PMON_CTR2_HIGH_REG.....	187
3.4.7 DCLK_PMON_CTR3_LOW_REG.....	187
3.4.8 DCLK_PMON_CTR3_HIGH_REG.....	187
3.4.9 DCLK_PMON_CTRCTL0_REG.....	187
3.4.10 DCLK_PMON_CTRCTL1_REG.....	188
3.4.11 DCLK_PMON_CTRCTL2_REG.....	189
3.4.12 DCLK_PMON_CTRCTL3_REG.....	190
3.4.13 DCLK_PMON_UNIT_CTL_REG.....	191
3.4.14 DCLK_PMON_UNIT_STATUS_REG.....	191
3.4.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	192
3.4.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	192
3.4.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	192
3.5 Bus: 2, Device: 9, Function: 2 (CFG).....	192
3.5.1 DCLK_PMON_CTR0_LOW_REG.....	193
3.5.2 DCLK_PMON_CTR0_HIGH_REG.....	193
3.5.3 DCLK_PMON_CTR1_LOW_REG.....	193
3.5.4 DCLK_PMON_CTR1_HIGH_REG.....	193
3.5.5 DCLK_PMON_CTR2_LOW_REG.....	194
3.5.6 DCLK_PMON_CTR2_HIGH_REG.....	194
3.5.7 DCLK_PMON_CTR3_LOW_REG.....	194
3.5.8 DCLK_PMON_CTR3_HIGH_REG.....	194
3.5.9 DCLK_PMON_CTRCTL0_REG.....	194
3.5.10 DCLK_PMON_CTRCTL1_REG.....	195
3.5.11 DCLK_PMON_CTRCTL2_REG.....	196
3.5.12 DCLK_PMON_CTRCTL3_REG.....	197
3.5.13 DCLK_PMON_UNIT_CTL_REG.....	198
3.5.14 DCLK_PMON_UNIT_STATUS_REG.....	199
3.5.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	199
3.5.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	199
3.5.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	200
3.6 Bus: 2, Device: 9, Function: 3 (CFG).....	200
3.6.1 DCLK_PMON_CTR0_LOW_REG.....	201
3.6.2 DCLK_PMON_CTR0_HIGH_REG.....	201
3.6.3 DCLK_PMON_CTR1_LOW_REG.....	201
3.6.4 DCLK_PMON_CTR1_HIGH_REG.....	201
3.6.5 DCLK_PMON_CTR2_LOW_REG.....	201
3.6.6 DCLK_PMON_CTR2_HIGH_REG.....	202
3.6.7 DCLK_PMON_CTR3_LOW_REG.....	202
3.6.8 DCLK_PMON_CTR3_HIGH_REG.....	202
3.6.9 DCLK_PMON_CTRCTL0_REG.....	202



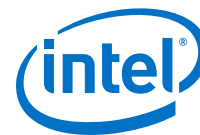
3.6.10 DCLK_PMON_CTL1_REG.....	203
3.6.11 DCLK_PMON_CTL2_REG.....	204
3.6.12 DCLK_PMON_CTL3_REG.....	205
3.6.13 DCLK_PMON_UNIT_CTL_REG.....	206
3.6.14 DCLK_PMON_UNIT_STATUS_REG.....	206
3.6.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	207
3.6.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	207
3.6.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	207
3.7 Bus: 2, Device: 9, Function: 4 (CFG).....	207
3.7.1 DCLK_PMON_CTR0_LOW_REG.....	208
3.7.2 DCLK_PMON_CTR0_HIGH_REG.....	208
3.7.3 DCLK_PMON_CTR1_LOW_REG.....	208
3.7.4 DCLK_PMON_CTR1_HIGH_REG.....	208
3.7.5 DCLK_PMON_CTR2_LOW_REG.....	209
3.7.6 DCLK_PMON_CTR2_HIGH_REG.....	209
3.7.7 DCLK_PMON_CTR3_LOW_REG.....	209
3.7.8 DCLK_PMON_CTR3_HIGH_REG.....	209
3.7.9 DCLK_PMON_CTL0_REG.....	209
3.7.10 DCLK_PMON_CTL1_REG.....	210
3.7.11 DCLK_PMON_CTL2_REG.....	211
3.7.12 DCLK_PMON_CTL3_REG.....	212
3.7.13 DCLK_PMON_UNIT_CTL_REG.....	213
3.7.14 DCLK_PMON_UNIT_STATUS_REG.....	214
3.7.15 DCLK_PMON_TIMESTAMP_LOW_REG.....	214
3.7.16 DCLK_PMON_TIMESTAMP_HIGH_REG.....	214
3.7.17 DCLK_PMON_TIMESTAMP_CTL_REG.....	215
3.8 Bus: 2, Device: 11, Function: 0 (CFG).....	215
3.8.1 UCLK_PMON_CTR0_LOW_REG.....	216
3.8.2 UCLK_PMON_CTR0_HIGH_REG.....	216
3.8.3 UCLK_PMON_CTR1_LOW_REG.....	216
3.8.4 UCLK_PMON_CTR1_HIGH_REG.....	216
3.8.5 UCLK_PMON_CTR2_LOW_REG.....	216
3.8.6 UCLK_PMON_CTR2_HIGH_REG.....	217
3.8.7 UCLK_PMON_CTR3_LOW_REG.....	217
3.8.8 UCLK_PMON_CTR3_HIGH_REG.....	217
3.8.9 UCLK_PMON_CTL0_REG.....	217
3.8.10 UCLK_PMON_CTL1_REG.....	218
3.8.11 UCLK_PMON_CTL2_REG.....	219
3.8.12 UCLK_PMON_CTL3_REG.....	220
3.8.13 UCLK_PMON_UNIT_CTL_REG.....	221
3.8.14 UCLK_PMON_UNIT_STATUS_REG.....	221
3.8.15 UCLK_PMON_TIMESTAMP_LOW_REG.....	222
3.8.16 UCLK_PMON_TIMESTAMP_HIGH_REG.....	222
3.8.17 UCLK_PMON_TIMESTAMP_CTL_REG.....	222
4.0 M2PCIe.....	223
4.1 Bus: 2, Device: 12, Function: 1 (CFG).....	223
4.1.1 PMONCNTRLOWER0_0.....	224
4.1.2 PMONCNTRUPPER0_0.....	224
4.1.3 PMONCNTRLOWER0_1.....	224
4.1.4 PMONCNTRUPPER0_1.....	224



4.1.5 PMONCNTRLOWER0_2.....	225
4.1.6 PMONCNTRUPPER0_2.....	225
4.1.7 PMONCNTRLOWER0_3.....	225
4.1.8 PMONCNTRUPPER0_3.....	225
4.1.9 PMONCNTRCFG0_0.....	225
4.1.10 PMONCNTRCFG0_1.....	226
4.1.11 PMONCNTRCFG0_2.....	227
4.1.12 PMONCNTRCFG0_3.....	228
4.1.13 PMONUNITCTRL0.....	229
4.1.14 PMONCTRSTATUS0.....	230
5.0 Model Specific Registers (MSR).....	231
5.1 Core MSRs.....	233
5.1.1 (C1h) IA32_PMC0.....	234
5.1.2 (C2h) IA32_PMC1.....	234
5.1.3 (186h) IA32_PERFEVTSEL0.....	234
5.1.4 (187h) IA32_PERFEVTSEL1.....	235
5.1.5 (1A0h) IA32_MISC_ENABLES.....	235
5.1.6 (1A6h) MSR_OFFCORE_RESP0.....	238
5.1.7 (1A7h) MSR_OFFCORE_RESP1.....	239
5.1.8 (1C8h) MSR_LBR_SELECT.....	240
5.1.9 (1C9h) MSR_LASTBRANCH_TOS.....	241
5.1.10 (1D9h) IA32_DEBUGCTL.....	241
5.1.11 (309h) IA32_FIXED_CTR0.....	242
5.1.12 (30Ah) IA32_FIXED_CTR1.....	242
5.1.13 (30Bh) IA32_FIXED_CTR2.....	242
5.1.14 (345h) IA32_PERF_CAPABILITIES.....	242
5.1.15 (38Dh) IA32_FIXED_CTR_CTRL.....	243
5.1.16 (38Eh) IA32_PERF_GLOBAL_STATUS.....	244
5.1.17 (38Fh) IA32_PERF_GLOBAL_CTRL.....	244
5.1.18 (390h) IA32_PERF_GLOBAL_OVF_CTRL.....	244
5.1.19 (3F1h) IA32_PEBS_ENABLE.....	245
5.1.20 (483h) IA32_VMX_EXIT_CTLS.....	245
5.1.21 (484h) IA32_VMX_ENTRY_CTLS.....	245
5.1.22 (4C1h) IA32_A_PMC0.....	245
5.1.23 (4C2h) IA32_A_PMC1.....	246
5.1.24 (680h) MSR_LASTBRANCH_0_FROM_IP.....	246
5.1.25 (681h) MSR_LASTBRANCH_1_FROM_IP.....	246
5.1.26 (682h) MSR_LASTBRANCH_2_FROM_IP.....	247
5.1.27 (683h) MSR_LASTBRANCH_3_FROM_IP.....	247
5.1.28 (684h) MSR_LASTBRANCH_4_FROM_IP.....	247
5.1.29 (685h) MSR_LASTBRANCH_5_FROM_IP.....	248
5.1.30 (686h) MSR_LASTBRANCH_6_FROM_IP.....	248
5.1.31 (687h) MSR_LASTBRANCH_7_FROM_IP.....	248
5.1.32 (6C0h) MSR_LASTBRANCH_0_TO_IP.....	249
5.1.33 (6C1h) MSR_LASTBRANCH_1_TO_IP.....	249
5.1.34 (6C2h) MSR_LASTBRANCH_2_TO_IP.....	249
5.1.35 (6C3h) MSR_LASTBRANCH_3_TO_IP.....	249
5.1.36 (6C4h) MSR_LASTBRANCH_4_TO_IP.....	250
5.1.37 (6C5h) MSR_LASTBRANCH_5_TO_IP.....	250
5.1.38 (6C6h) MSR_LASTBRANCH_6_TO_IP.....	250



5.1.39 (6C7h) MSR_LASTBRANCH_7_TO_IP.....	250
5.2 Uncore MSRs.....	251
5.2.1 (700h) NcuPMONGICtrl.....	264
5.2.2 (701h) NcuPMONGISts.....	269
5.2.3 (702h) NCUPMONConfig.....	272
5.2.4 (703h) NcuPMONCtlFix.....	272
5.2.5 (704h) NcuPMONCtrFx.....	273
5.2.6 (705h) NcuPMONCtl1.....	273
5.2.7 (706h) NcuPMONCtl2.....	274
5.2.8 (708h) NcuPMONCtSts.....	275
5.2.9 (709h) NcuPMONCt1.....	276
5.2.10 (70Ah) NcuPMONCt2.....	276
5.2.11 (710h) PmonUnitCtrl.....	276
5.2.12 (711h) PmonCntrCfg_0.....	277
5.2.13 (712h) PmonCntrCfg_1.....	279
5.2.14 (713h) PmonCntrCfg_2.....	281
5.2.15 (714h) PmonCntrCfg_3.....	283
5.2.16 (717h) PmonCntr_0.....	285
5.2.17 (718h) PmonCntr_1.....	285
5.2.18 (719h) PmonCntr_2.....	285
5.2.19 (71Ah) PmonCntr_3.....	286
5.2.20 (E00h) PERF_CTR_UNIT_CTRL_CHA_0.....	286
5.2.21 (E01h) PERF_EVT_SEL_0_CHA_0.....	287
5.2.22 (E02h) PERF_EVT_SEL_1_CHA_0.....	288
5.2.23 (E03h) PERF_EVT_SEL_2_CHA_0.....	289
5.2.24 (E04h) PERF_EVT_SEL_3_CHA_0.....	290
5.2.25 (E05h) PERF_UNIT_CTL_CHA_0.....	292
5.2.26 (E06h) PERF_UNIT_CTL_1_CHA_0.....	292
5.2.27 (E07h) PERF_UNIT_STATUS_CHA_0.....	293
5.2.28 (E08h) PERF_CTR_0_CHA_0.....	293
5.2.29 (E09h) PERF_CTR_1_CHA_0.....	293
5.2.30 (E0Ah) PERF_CTR_2_CHA_0.....	294
5.2.31 (E0Bh) PERF_CTR_3_CHA_0.....	294
5.2.32 (E0Ch) PERF_CTR_UNIT_CTRL_CHA_1.....	294
5.2.33 (E0Dh) PERF_EVT_SEL_0_CHA_1.....	295
5.2.34 (E0Eh) PERF_EVT_SEL_1_CHA_1.....	296
5.2.35 (E0Fh) PERF_EVT_SEL_2_CHA_1.....	297
5.2.36 (E10h) PERF_EVT_SEL_3_CHA_1.....	299
5.2.37 (E11h) PERF_UNIT_CTL_CHA_1.....	300
5.2.38 (E12h) PERF_UNIT_CTL_1_CHA_1.....	301
5.2.39 (E13h) PERF_UNIT_STATUS_CHA_1.....	301
5.2.40 (E14h) PERF_CTR_0_CHA_1.....	301
5.2.41 (E15h) PERF_CTR_1_CHA_1.....	302
5.2.42 (E16h) PERF_CTR_2_CHA_1.....	302
5.2.43 (E17h) PERF_CTR_3_CHA_1.....	302
5.2.44 (E18h) PERF_CTR_UNIT_CTRL_CHA_2.....	302
5.2.45 (E19h) PERF_EVT_SEL_0_CHA_2.....	303
5.2.46 (E1Ah) PERF_EVT_SEL_1_CHA_2.....	304
5.2.47 (E1Bh) PERF_EVT_SEL_2_CHA_2.....	306
5.2.48 (E1Ch) PERF_EVT_SEL_3_CHA_2.....	307
5.2.49 (E1Dh) PERF_UNIT_CTL_CHA_2.....	308



5.2.50 (E1Eh) PERF_UNIT_CTL_1_CHA_2.....	309
5.2.51 (E1Fh) PERF_UNIT_STATUS_CHA_2.....	309
5.2.52 (E20h) PERF_CTR_0_CHA_2.....	310
5.2.53 (E21h) PERF_CTR_1_CHA_2.....	310
5.2.54 (E22h) PERF_CTR_2_CHA_2.....	310
5.2.55 (E23h) PERF_CTR_3_CHA_2.....	310
5.2.56 (E24h) PERF_CTR_UNIT_CTRL_CHA_3.....	311
5.2.57 (E25h) PERF_EVT_SEL_0_CHA_3.....	311
5.2.58 (E26h) PERF_EVT_SEL_1_CHA_3.....	312
5.2.59 (E27h) PERF_EVT_SEL_2_CHA_3.....	314
5.2.60 (E28h) PERF_EVT_SEL_3_CHA_3.....	315
5.2.61 (E29h) PERF_UNIT_CTL_CHA_3.....	316
5.2.62 (E2Ah) PERF_UNIT_CTL_1_CHA_3.....	317
5.2.63 (E2Bh) PERF_UNIT_STATUS_CHA_3.....	318
5.2.64 (E2Ch) PERF_CTR_0_CHA_3.....	318
5.2.65 (E2Dh) PERF_CTR_1_CHA_3.....	318
5.2.66 (E2Eh) PERF_CTR_2_CHA_3.....	318
5.2.67 (E2Fh) PERF_CTR_3_CHA_3.....	318
5.2.68 (E30h) PERF_CTR_UNIT_CTRL_CHA_4.....	319
5.2.69 (E31h) PERF_EVT_SEL_0_CHA_4.....	319
5.2.70 (E32h) PERF_EVT_SEL_1_CHA_4.....	321
5.2.71 (E33h) PERF_EVT_SEL_2_CHA_4.....	322
5.2.72 (E34h) PERF_EVT_SEL_3_CHA_4.....	323
5.2.73 (E35h) PERF_UNIT_CTL_CHA_4.....	325
5.2.74 (E36h) PERF_UNIT_CTL_1_CHA_4.....	325
5.2.75 (E37h) PERF_UNIT_STATUS_CHA_4.....	326
5.2.76 (E38h) PERF_CTR_0_CHA_4.....	326
5.2.77 (E39h) PERF_CTR_1_CHA_4.....	326
5.2.78 (E3Ah) PERF_CTR_2_CHA_4.....	327
5.2.79 (E3Bh) PERF_CTR_3_CHA_4.....	327
5.2.80 (E3Ch) PERF_CTR_UNIT_CTRL_CHA_5.....	327
5.2.81 (E3Dh) PERF_EVT_SEL_0_CHA_5.....	328
5.2.82 (E3Eh) PERF_EVT_SEL_1_CHA_5.....	329
5.2.83 (E3Fh) PERF_EVT_SEL_2_CHA_5.....	330
5.2.84 (E40h) PERF_EVT_SEL_3_CHA_5.....	332
5.2.85 (E41h) PERF_UNIT_CTL_CHA_5.....	333
5.2.86 (E42h) PERF_UNIT_CTL_1_CHA_5.....	333
5.2.87 (E43h) PERF_UNIT_STATUS_CHA_5.....	334
5.2.88 (E44h) PERF_CTR_0_CHA_5.....	334
5.2.89 (E45h) PERF_CTR_1_CHA_5.....	335
5.2.90 (E46h) PERF_CTR_2_CHA_5.....	335
5.2.91 (E47h) PERF_CTR_3_CHA_5.....	335
5.2.92 (E48h) PERF_CTR_UNIT_CTRL_CHA_6.....	335
5.2.93 (E49h) PERF_EVT_SEL_0_CHA_6.....	336
5.2.94 (E4Ah) PERF_EVT_SEL_1_CHA_6.....	337
5.2.95 (E4Bh) PERF_EVT_SEL_2_CHA_6.....	338
5.2.96 (E4Ch) PERF_EVT_SEL_3_CHA_6.....	340
5.2.97 (E4Dh) PERF_UNIT_CTL_CHA_6.....	341
5.2.98 (E4Eh) PERF_UNIT_CTL_1_CHA_6.....	342
5.2.99 (E4Fh) PERF_UNIT_STATUS_CHA_6.....	342
5.2.100 (E50h) PERF_CTR_0_CHA_6.....	343



5.2.101 (E51h) PERF_CTR_1_CHA_6.....	343
5.2.102 (E52h) PERF_CTR_2_CHA_6.....	343
5.2.103 (E53h) PERF_CTR_3_CHA_6.....	343
5.2.104 (E54h) PERF_CTR_UNIT_CTRL_CHA_7.....	344
5.2.105 (E55h) PERF_EVT_SEL_0_CHA_7.....	344
5.2.106 (E56h) PERF_EVT_SEL_1_CHA_7.....	345
5.2.107 (E57h) PERF_EVT_SEL_2_CHA_7.....	347
5.2.108 (E58h) PERF_EVT_SEL_3_CHA_7.....	348
5.2.109 (E59h) PERF_UNIT_CTL_CHA_7.....	349
5.2.110 (E5Ah) PERF_UNIT_CTL_1_CHA_7.....	350
5.2.111 (E5Bh) PERF_UNIT_STATUS_CHA_7.....	351
5.2.112 (E5Ch) PERF_CTR_0_CHA_7.....	351
5.2.113 (E5Dh) PERF_CTR_1_CHA_7.....	351
5.2.114 (E5Eh) PERF_CTR_2_CHA_7.....	351
5.2.115 (E5Fh) PERF_CTR_3_CHA_7.....	351
5.2.116 (E60h) PERF_CTR_UNIT_CTRL_CHA_8.....	352
5.2.117 (E61h) PERF_EVT_SEL_0_CHA_8.....	352
5.2.118 (E62h) PERF_EVT_SEL_1_CHA_8.....	354
5.2.119 (E63h) PERF_EVT_SEL_2_CHA_8.....	355
5.2.120 (E64h) PERF_EVT_SEL_3_CHA_8.....	356
5.2.121 (E65h) PERF_UNIT_CTL_CHA_8.....	358
5.2.122 (E66h) PERF_UNIT_CTL_1_CHA_8.....	358
5.2.123 (E67h) PERF_UNIT_STATUS_CHA_8.....	359
5.2.124 (E68h) PERF_CTR_0_CHA_8.....	359
5.2.125 (E69h) PERF_CTR_1_CHA_8.....	359
5.2.126 (E6Ah) PERF_CTR_2_CHA_8.....	360
5.2.127 (E6Bh) PERF_CTR_3_CHA_8.....	360
5.2.128 (E6Ch) PERF_CTR_UNIT_CTRL_CHA_9.....	360
5.2.129 (E6Dh) PERF_EVT_SEL_0_CHA_9.....	361
5.2.130 (E6Eh) PERF_EVT_SEL_1_CHA_9.....	362
5.2.131 (E6Fh) PERF_EVT_SEL_2_CHA_9.....	363
5.2.132 (E70h) PERF_EVT_SEL_3_CHA_9.....	365
5.2.133 (E71h) PERF_UNIT_CTL_CHA_9.....	366
5.2.134 (E72h) PERF_UNIT_CTL_1_CHA_9.....	366
5.2.135 (E73h) PERF_UNIT_STATUS_CHA_9.....	367
5.2.136 (E74h) PERF_CTR_0_CHA_9.....	367
5.2.137 (E75h) PERF_CTR_1_CHA_9.....	368
5.2.138 (E76h) PERF_CTR_2_CHA_9.....	368
5.2.139 (E77h) PERF_CTR_3_CHA_9.....	368
5.2.140 (E78h) PERF_CTR_UNIT_CTRL_CHA_10.....	368
5.2.141 (E79h) PERF_EVT_SEL_0_CHA_10.....	369
5.2.142 (E7Ah) PERF_EVT_SEL_1_CHA_10.....	370
5.2.143 (E7Bh) PERF_EVT_SEL_2_CHA_10.....	371
5.2.144 (E7Ch) PERF_EVT_SEL_3_CHA_10.....	373
5.2.145 (E7Dh) PERF_UNIT_CTL_CHA_10.....	374
5.2.146 (E7Eh) PERF_UNIT_CTL_1_CHA_10.....	375
5.2.147 (E7Fh) PERF_UNIT_STATUS_CHA_10.....	375
5.2.148 (E80h) PERF_CTR_0_CHA_10.....	376
5.2.149 (E81h) PERF_CTR_1_CHA_10.....	376
5.2.150 (E82h) PERF_CTR_2_CHA_10.....	376
5.2.151 (E83h) PERF_CTR_3_CHA_10.....	376



5.2.152 (E84h) PERF_CTR_UNIT_CTRL_CHA_11.....	377
5.2.153 (E85h) PERF_EVT_SEL_0_CHA_11.....	377
5.2.154 (E86h) PERF_EVT_SEL_1_CHA_11.....	378
5.2.155 (E87h) PERF_EVT_SEL_2_CHA_11.....	380
5.2.156 (E88h) PERF_EVT_SEL_3_CHA_11.....	381
5.2.157 (E89h) PERF_UNIT_CTL_CHA_11.....	382
5.2.158 (E8Ah) PERF_UNIT_CTL_1_CHA_11.....	383
5.2.159 (E8Bh) PERF_UNIT_STATUS_CHA_11.....	384
5.2.160 (E8Ch) PERF_CTR_0_CHA_11.....	384
5.2.161 (E8Dh) PERF_CTR_1_CHA_11.....	384
5.2.162 (E8Eh) PERF_CTR_2_CHA_11.....	384
5.2.163 (E8Fh) PERF_CTR_3_CHA_11.....	384
5.2.164 (E90h) PERF_CTR_UNIT_CTRL_CHA_12.....	385
5.2.165 (E91h) PERF_EVT_SEL_0_CHA_12.....	385
5.2.166 (E92h) PERF_EVT_SEL_1_CHA_12.....	387
5.2.167 (E93h) PERF_EVT_SEL_2_CHA_12.....	388
5.2.168 (E94h) PERF_EVT_SEL_3_CHA_12.....	389
5.2.169 (E95h) PERF_UNIT_CTL_CHA_12.....	391
5.2.170 (E96h) PERF_UNIT_CTL_1_CHA_12.....	391
5.2.171 (E97h) PERF_UNIT_STATUS_CHA_12.....	392
5.2.172 (E98h) PERF_CTR_0_CHA_12.....	392
5.2.173 (E99h) PERF_CTR_1_CHA_12.....	392
5.2.174 (E9Ah) PERF_CTR_2_CHA_12.....	393
5.2.175 (E9Bh) PERF_CTR_3_CHA_12.....	393
5.2.176 (E9Ch) PERF_CTR_UNIT_CTRL_CHA_13.....	393
5.2.177 (E9Dh) PERF_EVT_SEL_0_CHA_13.....	394
5.2.178 (E9Eh) PERF_EVT_SEL_1_CHA_13.....	395
5.2.179 (E9Fh) PERF_EVT_SEL_2_CHA_13.....	396
5.2.180 (EA0h) PERF_EVT_SEL_3_CHA_13.....	398
5.2.181 (EA1h) PERF_UNIT_CTL_CHA_13.....	399
5.2.182 (EA2h) PERF_UNIT_CTL_1_CHA_13.....	399
5.2.183 (EA3h) PERF_UNIT_STATUS_CHA_13.....	400
5.2.184 (EA4h) PERF_CTR_0_CHA_13.....	400
5.2.185 (EA5h) PERF_CTR_1_CHA_13.....	401
5.2.186 (EA6h) PERF_CTR_2_CHA_13.....	401
5.2.187 (EA7h) PERF_CTR_3_CHA_13.....	401
5.2.188 (EA8h) PERF_CTR_UNIT_CTRL_CHA_14.....	401
5.2.189 (EA9h) PERF_EVT_SEL_0_CHA_14.....	402
5.2.190 (EAAh) PERF_EVT_SEL_1_CHA_14.....	403
5.2.191 (EABh) PERF_EVT_SEL_2_CHA_14.....	404
5.2.192 (EACH) PERF_EVT_SEL_3_CHA_14.....	406
5.2.193 (EADh) PERF_UNIT_CTL_CHA_14.....	407
5.2.194 (EAEh) PERF_UNIT_CTL_1_CHA_14.....	408
5.2.195 (EAFh) PERF_UNIT_STATUS_CHA_14.....	408
5.2.196 (EB0h) PERF_CTR_0_CHA_14.....	409
5.2.197 (EB1h) PERF_CTR_1_CHA_14.....	409
5.2.198 (EB2h) PERF_CTR_2_CHA_14.....	409
5.2.199 (EB3h) PERF_CTR_3_CHA_14.....	409
5.2.200 (EB4h) PERF_CTR_UNIT_CTRL_CHA_15.....	410
5.2.201 (EB5h) PERF_EVT_SEL_0_CHA_15.....	410
5.2.202 (EB6h) PERF_EVT_SEL_1_CHA_15.....	411



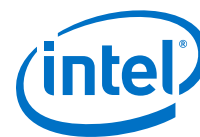
5.2.203 (EB7h) PERF_EVT_SEL_2_CHA_15.....	413
5.2.204 (EB8h) PERF_EVT_SEL_3_CHA_15.....	414
5.2.205 (EB9h) PERF_UNIT_CTL_CHA_15.....	415
5.2.206 (EBAh) PERF_UNIT_CTL_1_CHA_15.....	416
5.2.207 (EBBh) PERF_UNIT_STATUS_CHA_15.....	417
5.2.208 (EBCh) PERF_CTR_0_CHA_15.....	417
5.2.209 (EBDh) PERF_CTR_1_CHA_15.....	417
5.2.210 (EBEh) PERF_CTR_2_CHA_15.....	417
5.2.211 (EBFh) PERF_CTR_3_CHA_15.....	417
5.2.212 (EC0h) PERF_CTR_UNIT_CTRL_CHA_16.....	418
5.2.213 (EC1h) PERF_EVT_SEL_0_CHA_16.....	418
5.2.214 (EC2h) PERF_EVT_SEL_1_CHA_16.....	420
5.2.215 (EC3h) PERF_EVT_SEL_2_CHA_16.....	421
5.2.216 (EC4h) PERF_EVT_SEL_3_CHA_16.....	422
5.2.217 (EC5h) PERF_UNIT_CTL_CHA_16.....	424
5.2.218 (EC6h) PERF_UNIT_CTL_1_CHA_16.....	424
5.2.219 (EC7h) PERF_UNIT_STATUS_CHA_16.....	425
5.2.220 (EC8h) PERF_CTR_0_CHA_16.....	425
5.2.221 (EC9h) PERF_CTR_1_CHA_16.....	425
5.2.222 (ECAh) PERF_CTR_2_CHA_16.....	426
5.2.223 (ECBh) PERF_CTR_3_CHA_16.....	426
5.2.224 (ECCh) PERF_CTR_UNIT_CTRL_CHA_17.....	426
5.2.225 (ECDh) PERF_EVT_SEL_0_CHA_17.....	427
5.2.226 (ECEh) PERF_EVT_SEL_1_CHA_17.....	428
5.2.227 (ECFh) PERF_EVT_SEL_2_CHA_17.....	429
5.2.228 (ED0h) PERF_EVT_SEL_3_CHA_17.....	431
5.2.229 (ED1h) PERF_UNIT_CTL_CHA_17.....	432
5.2.230 (ED2h) PERF_UNIT_CTL_1_CHA_17.....	432
5.2.231 (ED3h) PERF_UNIT_STATUS_CHA_17.....	433
5.2.232 (ED4h) PERF_CTR_0_CHA_17.....	433
5.2.233 (ED5h) PERF_CTR_1_CHA_17.....	434
5.2.234 (ED6h) PERF_CTR_2_CHA_17.....	434
5.2.235 (ED7h) PERF_CTR_3_CHA_17.....	434
5.2.236 (ED8h) PERF_CTR_UNIT_CTRL_CHA_18.....	434
5.2.237 (ED9h) PERF_EVT_SEL_0_CHA_18.....	435
5.2.238 (EDAh) PERF_EVT_SEL_1_CHA_18.....	436
5.2.239 (EDBh) PERF_EVT_SEL_2_CHA_18.....	437
5.2.240 (EDCh) PERF_EVT_SEL_3_CHA_18.....	439
5.2.241 (EDDh) PERF_UNIT_CTL_CHA_18.....	440
5.2.242 (EDEh) PERF_UNIT_CTL_1_CHA_18.....	441
5.2.243 (EDFh) PERF_UNIT_STATUS_CHA_18.....	441
5.2.244 (EE0h) PERF_CTR_0_CHA_18.....	442
5.2.245 (EE1h) PERF_CTR_1_CHA_18.....	442
5.2.246 (EE2h) PERF_CTR_2_CHA_18.....	442
5.2.247 (EE3h) PERF_CTR_3_CHA_18.....	442
5.2.248 (EE4h) PERF_CTR_UNIT_CTRL_CHA_19.....	443
5.2.249 (EE5h) PERF_EVT_SEL_0_CHA_19.....	443
5.2.250 (EE6h) PERF_EVT_SEL_1_CHA_19.....	444
5.2.251 (EE7h) PERF_EVT_SEL_2_CHA_19.....	446
5.2.252 (EE8h) PERF_EVT_SEL_3_CHA_19.....	447
5.2.253 (EE9h) PERF_UNIT_CTL_CHA_19.....	448



5.2.254 (EEAh) PERF_UNIT_CTL_1_CHA_19.....	449
5.2.255 (EEBh) PERF_UNIT_STATUS_CHA_19.....	450
5.2.256 (EECh) PERF_CTR_0_CHA_19.....	450
5.2.257 (EEDh) PERF_CTR_1_CHA_19.....	450
5.2.258 (EEEh) PERF_CTR_2_CHA_19.....	450
5.2.259 (EEFh) PERF_CTR_3_CHA_19.....	450
5.2.260 (EF0h) PERF_CTR_UNIT_CTRL_CHA_20.....	451
5.2.261 (EF1h) PERF_EVT_SEL_0_CHA_20.....	451
5.2.262 (EF2h) PERF_EVT_SEL_1_CHA_20.....	453
5.2.263 (EF3h) PERF_EVT_SEL_2_CHA_20.....	454
5.2.264 (EF4h) PERF_EVT_SEL_3_CHA_20.....	455
5.2.265 (EF5h) PERF_UNIT_CTL_CHA_20.....	457
5.2.266 (EF6h) PERF_UNIT_CTL_1_CHA_20.....	457
5.2.267 (EF7h) PERF_UNIT_STATUS_CHA_20.....	458
5.2.268 (EF8h) PERF_CTR_0_CHA_20.....	458
5.2.269 (EF9h) PERF_CTR_1_CHA_20.....	458
5.2.270 (EFAh) PERF_CTR_2_CHA_20.....	459
5.2.271 (EFBh) PERF_CTR_3_CHA_20.....	459
5.2.272 (EFCh) PERF_CTR_UNIT_CTRL_CHA_21.....	459
5.2.273 (EFDh) PERF_EVT_SEL_0_CHA_21.....	460
5.2.274 (EFEh) PERF_EVT_SEL_1_CHA_21.....	461
5.2.275 (EFFh) PERF_EVT_SEL_2_CHA_21.....	462
5.2.276 (F00h) PERF_EVT_SEL_3_CHA_21.....	464
5.2.277 (F01h) PERF_UNIT_CTL_CHA_21.....	465
5.2.278 (F02h) PERF_UNIT_CTL_1_CHA_21.....	465
5.2.279 (F03h) PERF_UNIT_STATUS_CHA_21.....	466
5.2.280 (F04h) PERF_CTR_0_CHA_21.....	466
5.2.281 (F05h) PERF_CTR_1_CHA_21.....	467
5.2.282 (F06h) PERF_CTR_2_CHA_21.....	467
5.2.283 (F07h) PERF_CTR_3_CHA_21.....	467
5.2.284 (F08h) PERF_CTR_UNIT_CTRL_CHA_22.....	467
5.2.285 (F09h) PERF_EVT_SEL_0_CHA_22.....	468
5.2.286 (F0Ah) PERF_EVT_SEL_1_CHA_22.....	469
5.2.287 (F0Bh) PERF_EVT_SEL_2_CHA_22.....	470
5.2.288 (F0Ch) PERF_EVT_SEL_3_CHA_22.....	472
5.2.289 (F0Dh) PERF_UNIT_CTL_CHA_22.....	473
5.2.290 (F0Eh) PERF_UNIT_CTL_1_CHA_22.....	474
5.2.291 (F0Fh) PERF_UNIT_STATUS_CHA_22.....	474
5.2.292 (F10h) PERF_CTR_0_CHA_22.....	475
5.2.293 (F11h) PERF_CTR_1_CHA_22.....	475
5.2.294 (F12h) PERF_CTR_2_CHA_22.....	475
5.2.295 (F13h) PERF_CTR_3_CHA_22.....	475
5.2.296 (F14h) PERF_CTR_UNIT_CTRL_CHA_23.....	476
5.2.297 (F15h) PERF_EVT_SEL_0_CHA_23.....	476
5.2.298 (F16h) PERF_EVT_SEL_1_CHA_23.....	477
5.2.299 (F17h) PERF_EVT_SEL_2_CHA_23.....	479
5.2.300 (F18h) PERF_EVT_SEL_3_CHA_23.....	480
5.2.301 (F19h) PERF_UNIT_CTL_CHA_23.....	481
5.2.302 (F1Ah) PERF_UNIT_CTL_1_CHA_23.....	482
5.2.303 (F1Bh) PERF_UNIT_STATUS_CHA_23.....	483
5.2.304 (F1Ch) PERF_CTR_0_CHA_23.....	483



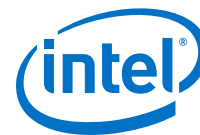
5.2.305 (F1Dh) PERF_CTR_1_CHA_23.....	483
5.2.306 (F1Eh) PERF_CTR_2_CHA_23.....	483
5.2.307 (F1Fh) PERF_CTR_3_CHA_23.....	483
5.2.308 (F20h) PERF_CTR_UNIT_CTRL_CHA_24.....	484
5.2.309 (F21h) PERF_EVT_SEL_0_CHA_24.....	484
5.2.310 (F22h) PERF_EVT_SEL_1_CHA_24.....	486
5.2.311 (F23h) PERF_EVT_SEL_2_CHA_24.....	487
5.2.312 (F24h) PERF_EVT_SEL_3_CHA_24.....	488
5.2.313 (F25h) PERF_UNIT_CTL_CHA_24.....	490
5.2.314 (F26h) PERF_UNIT_CTL_1_CHA_24.....	490
5.2.315 (F27h) PERF_UNIT_STATUS_CHA_24.....	491
5.2.316 (F28h) PERF_CTR_0_CHA_24.....	491
5.2.317 (F29h) PERF_CTR_1_CHA_24.....	491
5.2.318 (F2Ah) PERF_CTR_2_CHA_24.....	492
5.2.319 (F2Bh) PERF_CTR_3_CHA_24.....	492
5.2.320 (F2Ch) PERF_CTR_UNIT_CTRL_CHA_25.....	492
5.2.321 (F2Dh) PERF_EVT_SEL_0_CHA_25.....	493
5.2.322 (F2Eh) PERF_EVT_SEL_1_CHA_25.....	494
5.2.323 (F2Fh) PERF_EVT_SEL_2_CHA_25.....	495
5.2.324 (F30h) PERF_EVT_SEL_3_CHA_25.....	497
5.2.325 (F31h) PERF_UNIT_CTL_CHA_25.....	498
5.2.326 (F32h) PERF_UNIT_CTL_1_CHA_25.....	498
5.2.327 (F33h) PERF_UNIT_STATUS_CHA_25.....	499
5.2.328 (F34h) PERF_CTR_0_CHA_25.....	499
5.2.329 (F35h) PERF_CTR_1_CHA_25.....	500
5.2.330 (F36h) PERF_CTR_2_CHA_25.....	500
5.2.331 (F37h) PERF_CTR_3_CHA_25.....	500
5.2.332 (F38h) PERF_CTR_UNIT_CTRL_CHA_26.....	500
5.2.333 (F39h) PERF_EVT_SEL_0_CHA_26.....	501
5.2.334 (F3Ah) PERF_EVT_SEL_1_CHA_26.....	502
5.2.335 (F3Bh) PERF_EVT_SEL_2_CHA_26.....	503
5.2.336 (F3Ch) PERF_EVT_SEL_3_CHA_26.....	505
5.2.337 (F3Dh) PERF_UNIT_CTL_CHA_26.....	506
5.2.338 (F3Eh) PERF_UNIT_CTL_1_CHA_26.....	507
5.2.339 (F3Fh) PERF_UNIT_STATUS_CHA_26.....	507
5.2.340 (F40h) PERF_CTR_0_CHA_26.....	508
5.2.341 (F41h) PERF_CTR_1_CHA_26.....	508
5.2.342 (F42h) PERF_CTR_2_CHA_26.....	508
5.2.343 (F43h) PERF_CTR_3_CHA_26.....	508
5.2.344 (F44h) PERF_CTR_UNIT_CTRL_CHA_27.....	509
5.2.345 (F45h) PERF_EVT_SEL_0_CHA_27.....	509
5.2.346 (F46h) PERF_EVT_SEL_1_CHA_27.....	510
5.2.347 (F47h) PERF_EVT_SEL_2_CHA_27.....	512
5.2.348 (F48h) PERF_EVT_SEL_3_CHA_27.....	513
5.2.349 (F49h) PERF_UNIT_CTL_CHA_27.....	514
5.2.350 (F4Ah) PERF_UNIT_CTL_1_CHA_27.....	515
5.2.351 (F4Bh) PERF_UNIT_STATUS_CHA_27.....	516
5.2.352 (F4Ch) PERF_CTR_0_CHA_27.....	516
5.2.353 (F4Dh) PERF_CTR_1_CHA_27.....	516
5.2.354 (F4Eh) PERF_CTR_2_CHA_27.....	516
5.2.355 (F4Fh) PERF_CTR_3_CHA_27.....	516



5.2.356 (F50h) PERF_CTR_UNIT_CTRL_CHA_28.....	517
5.2.357 (F51h) PERF_EVT_SEL_0_CHA_28.....	517
5.2.358 (F52h) PERF_EVT_SEL_1_CHA_28.....	519
5.2.359 (F53h) PERF_EVT_SEL_2_CHA_28.....	520
5.2.360 (F54h) PERF_EVT_SEL_3_CHA_28.....	521
5.2.361 (F55h) PERF_UNIT_CTL_CHA_28.....	523
5.2.362 (F56h) PERF_UNIT_CTL_1_CHA_28.....	523
5.2.363 (F57h) PERF_UNIT_STATUS_CHA_28.....	524
5.2.364 (F58h) PERF_CTR_0_CHA_28.....	524
5.2.365 (F59h) PERF_CTR_1_CHA_28.....	524
5.2.366 (F5Ah) PERF_CTR_2_CHA_28.....	525
5.2.367 (F5Bh) PERF_CTR_3_CHA_28.....	525
5.2.368 (F5Ch) PERF_CTR_UNIT_CTRL_CHA_29.....	525
5.2.369 (F5Dh) PERF_EVT_SEL_0_CHA_29.....	526
5.2.370 (F5Eh) PERF_EVT_SEL_1_CHA_29.....	527
5.2.371 (F5Fh) PERF_EVT_SEL_2_CHA_29.....	528
5.2.372 (F60h) PERF_EVT_SEL_3_CHA_29.....	530
5.2.373 (F61h) PERF_UNIT_CTL_CHA_29.....	531
5.2.374 (F62h) PERF_UNIT_CTL_1_CHA_29.....	531
5.2.375 (F63h) PERF_UNIT_STATUS_CHA_29.....	532
5.2.376 (F64h) PERF_CTR_0_CHA_29.....	532
5.2.377 (F65h) PERF_CTR_1_CHA_29.....	533
5.2.378 (F66h) PERF_CTR_2_CHA_29.....	533
5.2.379 (F67h) PERF_CTR_3_CHA_29.....	533
5.2.380 (F68h) PERF_CTR_UNIT_CTRL_CHA_30.....	533
5.2.381 (F69h) PERF_EVT_SEL_0_CHA_30.....	534
5.2.382 (F6Ah) PERF_EVT_SEL_1_CHA_30.....	535
5.2.383 (F6Bh) PERF_EVT_SEL_2_CHA_30.....	536
5.2.384 (F6Ch) PERF_EVT_SEL_3_CHA_30.....	538
5.2.385 (F6Dh) PERF_UNIT_CTL_CHA_30.....	539
5.2.386 (F6Eh) PERF_UNIT_CTL_1_CHA_30.....	540
5.2.387 (F6Fh) PERF_UNIT_STATUS_CHA_30.....	540
5.2.388 (F70h) PERF_CTR_0_CHA_30.....	541
5.2.389 (F71h) PERF_CTR_1_CHA_30.....	541
5.2.390 (F72h) PERF_CTR_2_CHA_30.....	541
5.2.391 (F73h) PERF_CTR_3_CHA_30.....	541
5.2.392 (F74h) PERF_CTR_UNIT_CTRL_CHA_31.....	542
5.2.393 (F75h) PERF_EVT_SEL_0_CHA_31.....	542
5.2.394 (F76h) PERF_EVT_SEL_1_CHA_31.....	543
5.2.395 (F77h) PERF_EVT_SEL_2_CHA_31.....	545
5.2.396 (F78h) PERF_EVT_SEL_3_CHA_31.....	546
5.2.397 (F79h) PERF_UNIT_CTL_CHA_31.....	547
5.2.398 (F7Ah) PERF_UNIT_CTL_1_CHA_31.....	548
5.2.399 (F7Bh) PERF_UNIT_STATUS_CHA_31.....	549
5.2.400 (F7Ch) PERF_CTR_0_CHA_31.....	549
5.2.401 (F7Dh) PERF_CTR_1_CHA_31.....	549
5.2.402 (F7Eh) PERF_CTR_2_CHA_31.....	549
5.2.403 (F7Fh) PERF_CTR_3_CHA_31.....	549
5.2.404 (F80h) PERF_CTR_UNIT_CTRL_CHA_32.....	550
5.2.405 (F81h) PERF_EVT_SEL_0_CHA_32.....	550
5.2.406 (F82h) PERF_EVT_SEL_1_CHA_32.....	552



5.2.407 (F83h) PERF_EVT_SEL_2_CHA_32.....	553
5.2.408 (F84h) PERF_EVT_SEL_3_CHA_32.....	554
5.2.409 (F85h) PERF_UNIT_CTL_CHA_32.....	556
5.2.410 (F86h) PERF_UNIT_CTL_1_CHA_32.....	556
5.2.411 (F87h) PERF_UNIT_STATUS_CHA_32.....	557
5.2.412 (F88h) PERF_CTR_0_CHA_32.....	557
5.2.413 (F89h) PERF_CTR_1_CHA_32.....	557
5.2.414 (F8Ah) PERF_CTR_2_CHA_32.....	558
5.2.415 (F8Bh) PERF_CTR_3_CHA_32.....	558
5.2.416 (F8Ch) PERF_CTR_UNIT_CTRL_CHA_33.....	558
5.2.417 (F8Dh) PERF_EVT_SEL_0_CHA_33.....	559
5.2.418 (F8Eh) PERF_EVT_SEL_1_CHA_33.....	560
5.2.419 (F8Fh) PERF_EVT_SEL_2_CHA_33.....	561
5.2.420 (F90h) PERF_EVT_SEL_3_CHA_33.....	563
5.2.421 (F91h) PERF_UNIT_CTL_CHA_33.....	564
5.2.422 (F92h) PERF_UNIT_CTL_1_CHA_33.....	564
5.2.423 (F93h) PERF_UNIT_STATUS_CHA_33.....	565
5.2.424 (F94h) PERF_CTR_0_CHA_33.....	565
5.2.425 (F95h) PERF_CTR_1_CHA_33.....	566
5.2.426 (F96h) PERF_CTR_2_CHA_33.....	566
5.2.427 (F97h) PERF_CTR_3_CHA_33.....	566
5.2.428 (F98h) PERF_CTR_UNIT_CTRL_CHA_34.....	566
5.2.429 (F99h) PERF_EVT_SEL_0_CHA_34.....	567
5.2.430 (F9Ah) PERF_EVT_SEL_1_CHA_34.....	568
5.2.431 (F9Bh) PERF_EVT_SEL_2_CHA_34.....	569
5.2.432 (F9Ch) PERF_EVT_SEL_3_CHA_34.....	571
5.2.433 (F9Dh) PERF_UNIT_CTL_CHA_34.....	572
5.2.434 (F9Eh) PERF_UNIT_CTL_1_CHA_34.....	573
5.2.435 (F9Fh) PERF_UNIT_STATUS_CHA_34.....	573
5.2.436 (FA0h) PERF_CTR_0_CHA_34.....	574
5.2.437 (FA1h) PERF_CTR_1_CHA_34.....	574
5.2.438 (FA2h) PERF_CTR_2_CHA_34.....	574
5.2.439 (FA3h) PERF_CTR_3_CHA_34.....	574
5.2.440 (FA4h) PERF_CTR_UNIT_CTRL_CHA_35.....	575
5.2.441 (FA5h) PERF_EVT_SEL_0_CHA_35.....	575
5.2.442 (FA6h) PERF_EVT_SEL_1_CHA_35.....	576
5.2.443 (FA7h) PERF_EVT_SEL_2_CHA_35.....	578
5.2.444 (FA8h) PERF_EVT_SEL_3_CHA_35.....	579
5.2.445 (FA9h) PERF_UNIT_CTL_CHA_35.....	580
5.2.446 (FAAh) PERF_UNIT_CTL_1_CHA_35.....	581
5.2.447 (FABh) PERF_UNIT_STATUS_CHA_35.....	582
5.2.448 (FACH) PERF_CTR_0_CHA_35.....	582
5.2.449 (FADh) PERF_CTR_1_CHA_35.....	582
5.2.450 (FAEh) PERF_CTR_2_CHA_35.....	582
5.2.451 (FAFh) PERF_CTR_3_CHA_35.....	582
5.2.452 (FB0h) PERF_CTR_UNIT_CTRL_CHA_36.....	583
5.2.453 (FB1h) PERF_EVT_SEL_0_CHA_36.....	583
5.2.454 (FB2h) PERF_EVT_SEL_1_CHA_36.....	585
5.2.455 (FB3h) PERF_EVT_SEL_2_CHA_36.....	586
5.2.456 (FB4h) PERF_EVT_SEL_3_CHA_36.....	587
5.2.457 (FB5h) PERF_UNIT_CTL_CHA_36.....	589



5.2.458 (FB6h) PERF_UNIT_CTL_1_CHA_36.....	589
5.2.459 (FB7h) PERF_UNIT_STATUS_CHA_36.....	590
5.2.460 (FB8h) PERF_CTR_0_CHA_36.....	590
5.2.461 (FB9h) PERF_CTR_1_CHA_36.....	590
5.2.462 (FBAh) PERF_CTR_2_CHA_36.....	591
5.2.463 (FBBh) PERF_CTR_3_CHA_36.....	591
5.2.464 (FBCh) PERF_CTR_UNIT_CTRL_CHA_37.....	591
5.2.465 (FBDh) PERF_EVT_SEL_0_CHA_37.....	592
5.2.466 (FBEh) PERF_EVT_SEL_1_CHA_37.....	593
5.2.467 (FBFh) PERF_EVT_SEL_2_CHA_37.....	594
5.2.468 (FC0h) PERF_EVT_SEL_3_CHA_37.....	596
5.2.469 (FC1h) PERF_UNIT_CTL_CHA_37.....	597
5.2.470 (FC2h) PERF_UNIT_CTL_1_CHA_37.....	597
5.2.471 (FC3h) PERF_UNIT_STATUS_CHA_37.....	598
5.2.472 (FC4h) PERF_CTR_0_CHA_37.....	598
5.2.473 (FC5h) PERF_CTR_1_CHA_37.....	599
5.2.474 (FC6h) PERF_CTR_2_CHA_37.....	599
5.2.475 (FC7h) PERF_CTR_3_CHA_37.....	599
6.0 IRP.....	600
6.1 IRP0_PMONCNTR_0: PMON Counter.....	600
6.2 IRP0_PMONCNTR_1: PMON Counter.....	600
6.3 IRP0_PMONCNTRCFG_0: Counter Config.....	600
6.4 IRP0_PMONCNTRCFG_1: Counter Config.....	602
6.5 IRP_PMONUNITCTRL: Unit Control.....	603
6.6 IRP_PMONUNITSTATUS: Unit Status.....	604



Figures

1	The processor Tile Block Diagram	27
2	The processor Overview.....	27
3	Untile PMON module unit control register format	33
4	Untile PMON module unit status register format	33
5	Counter Control register that pairs with each counter	33



Tables

1	PEBS Performance Events for the processor	30
2	PEBS Record Format for the processor.....	31
3	OffCore Response Event Encoding.....	31
4	The processor Per-Component Performance Monitoring Capabilities.....	32
5	Glossary	34
6	Register Attribute Definitions.....	36
7	EDC0 Register ID/Name Mapping.....	39
8	Summary of Bus: 2, Device: 15, Function: 0 (CFG).....	40
9	Summary of Bus: 2, Device: 16, Function: 0 (CFG).....	48
10	Summary of Bus: 2, Device: 17, Function: 0 (CFG).....	55
11	Summary of Bus: 2, Device: 18, Function: 0 (CFG).....	63
12	Summary of Bus: 2, Device: 19, Function: 0 (CFG).....	70
13	Summary of Bus: 2, Device: 20, Function: 0 (CFG).....	78
14	Summary of Bus: 2, Device: 21, Function: 0 (CFG).....	85
15	Summary of Bus: 2, Device: 22, Function: 0 (CFG).....	93
16	Summary of Bus: 2, Device: 24, Function: 2 (CFG).....	100
17	Summary of Bus: 2, Device: 25, Function: 2 (CFG).....	108
18	Summary of Bus: 2, Device: 26, Function: 2 (CFG).....	115
19	Summary of Bus: 2, Device: 27, Function: 2 (CFG).....	123
20	Summary of Bus: 2, Device: 28, Function: 2 (CFG).....	130
21	Summary of Bus: 2, Device: 29, Function: 2 (CFG).....	138
22	Summary of Bus: 2, Device: 30, Function: 2 (CFG).....	145
23	Summary of Bus: 2, Device: 31, Function: 2 (CFG).....	153
24	MC0 Register ID/Name Mapping.....	161
25	Summary of Bus: 2, Device: 10, Function: 0 (CFG).....	162
26	Summary of Bus: 2, Device: 8, Function: 2 (CFG).....	170
27	Summary of Bus: 2, Device: 8, Function: 3 (CFG).....	177
28	Summary of Bus: 2, Device: 8, Function: 4 (CFG).....	185
29	Summary of Bus: 2, Device: 9, Function: 2 (CFG).....	192
30	Summary of Bus: 2, Device: 9, Function: 3 (CFG).....	200
31	Summary of Bus: 2, Device: 9, Function: 4 (CFG).....	207
32	Summary of Bus: 2, Device: 11, Function: 0 (CFG).....	215
33	M2PCIE Register ID/Name Mapping.....	223
34	Summary of Bus: 2, Device: 12, Function: 1 (CFG).....	223
35	UBOX and CHA0 Register ID/Name Mapping.....	232
36	Summary of Core MSR Registers.....	233
37	Summary of Uncore MSR Registers.....	251
38	Summary of Bus:0 Device:5 Function:6.....	600



1.0 Performance Monitoring in the Intel® Xeon Phi™ processor - Introduction

The Intel® Xeon Phi™ processor (formerly code named Knights Landing, also referred to as "the processor" in the document) provides facilities for monitoring performance. The processor's performance monitoring capabilities are distributed throughout its tile and untile topology. It is a unique combination of Intel® Atom™ processor like core Performance monitoring unit (PMU) capabilities and Intel® Xeon® processor based server class uncore capabilities.

The *Intel® Xeon Phi™ processor - Performance monitoring reference manual Volume 1: Registers* document provides Tile and Untile register and MSR information related to Performance Monitoring on the processor. This document is targeted towards software developers involved in developing device drivers and software agents that wish to perform performance monitoring in the processor. Please note that the reader might notice a difference in the bus mapping as a result of using a different BIOS. The reader is expected to be familiar with other industry standards and specifications listed in the remainder of this section.

This document is to be used in conjunction with the Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume 2 : Events document located [here](#).

Before looking into the performance monitoring capabilities offered, the following section gives a brief overview of the processor architecture.

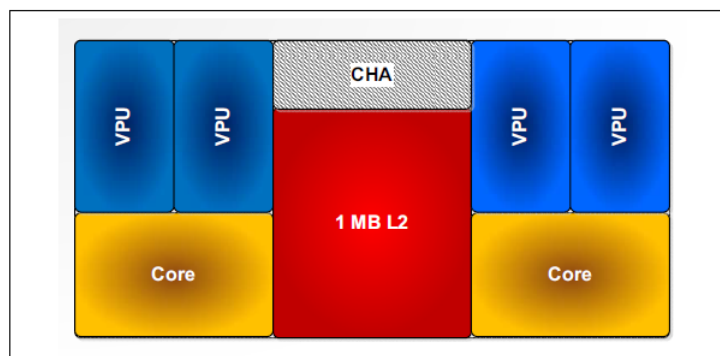
1.1 Intel® Xeon Phi™ Processor Architecture Overview

The second generation Intel® Xeon Phi™ processor is based on the Intel® Many Integrated Core (MIC) architecture built on 14-nanometer process technology. This product family is designed for High Performance Computing (HPC) and is optimized for highly parallel applications.

The processor has up to 72 Cores with each core supporting 4 threads giving a total of 288 threads in a socket. The Cores are organized as Tiles. Each Tile contains 2 Cores, 4 Vector Processing Units (VPU), L2 Cache and CHA. So there are total of 36 Tiles in the processor CPU.

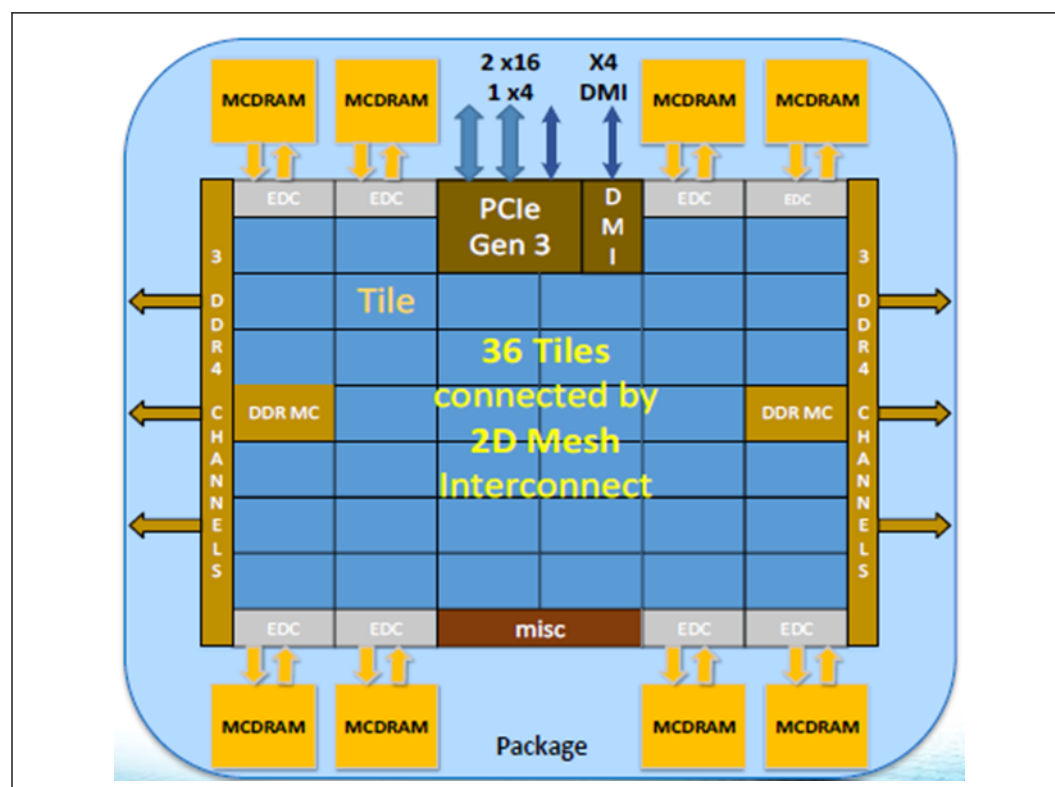


Figure 1. The processor Tile Block Diagram



The processor Untile is an LLC-less design that uses the mesh architecture to interconnect all the Tiles and thus the Cores. The processor is the first CPU to use the mesh architecture. It also integrates the Caching home agent (CHA) in a distributed caching architecture.

Figure 2. The processor Overview



The processor supports DDR4 memory technology with a total of six channels, 3 per memory controller (1 DIMM per channel only. Up to 64 GB per DIMM; up to 384 GB total].



It also introduces new memory technology called Multi Channel DRAM (MCDRAM) into the server space. MCDRAM is a high band width memory designed to meet the bandwidth demand due to high number of threads present in the Knights Landing Processor. MCDRAM is an on-package memory that is connected to the processor's Embedded DRAM Controller (EDC) and has a maximum capacity of 16GB.

The processor also has Integrated IO (IIO) which supports x4 Gen2 DMI connection to PCH and up to x36 Gen3 PCIe lanes.

The processor Untile feature summary:

- Memory:
 1. DDR4: 6 channels @ 2400 up to 384GB.
 2. MCDRAM: Upto 16 GB on-package; High BW.
- Two integrated memory controllers (IMCs) each supporting three DDR4 channels. IMCs provide the interface to DDR4 DIMMs and communicates to the rest of the processor via the Mesh interconnect.
- Eight embedded DRAM controllers (EDCs) which is the high bandwidth near-memory controller for the MCDRAM interface with the on-package MCDRAM die.
- One IIO module to interface to 36 Gen3 PCIe lanes and 4 DMI2 lanes.
- A 2D mesh which is a high bandwidth interconnect between the tile (upto 36) and untile modules.
- The utility box (Ubox) sharing a mesh stop with the IIO.
- The Caching Agent (CHA) provides the horizontal and vertical linkage of all clusters across the mesh.
- Power Control Unit (PCU) microcontroller provides power and thermal management for the processor.

1.1.1 Intel® Xeon Phi™ processor Memory and Cluster Modes

The processor supports three MCDRAM memory configuration modes namely Cache Mode, Flat Mode and Hybrid Mode.

- **Cache:** In Cache mode, the MCDRAM will act like a memory side cache (MSC) for the DDR4 memory. MCDRAM is used as a direct-mapped, memory-side, 64B cacheline, cache where the tag is stored in the MCDRAM.
- **Flat:** Physical address treats all MCDRAM and DDR4 as one memory space. The degenerate modes with only MCDRAM, or only DDR4, are supported.
- **Hybrid:** Allows part of the MCDRAM capacity to be used for cache and the remaining capacity to be used as memory

In addition, the processor also supports three cluster configuration modes. These configurations determine how the tiles align with the memory.

- **All-to-all:** Memory traffic addresses are uniformly hashed across all distributed directories. This mode is only required if non-uniformly configured DDR4 DIMMs are loaded in the system.
- **Quadrant:** The tile array is divided into four quadrants, and the directory for memory traffic addresses resides in the same quadrant as the memory location. This is the default configuration of the processor, and is transparent to software.



- **Sub-NUMA:** The tile array is divided into four quadrants, each configured as a separate NUMA domain to the OS. This optional configuration requires all 36 tiles to be enabled.

1.2 Performance Monitoring in the Intel® Xeon Phi™ processor tile

A tile is a core-pair similar to those found in Intel® Atom™ processors based on the Silvermont microarchitecture (referred to as "SLM" in the rest of the document), but added HyperThreading Technology and additional enhancements. The Processor provides several new capabilities on top of the SLM performance monitoring facilities. For more information on performance monitoring capabilities in SLM, refer to the Section 18.6 in the Intel® 64 and IA-32 Architectures Software Developer Manuals (SDM).

The processor supports architectural performance monitoring capability with version ID 3 (see Section 18.2.3 in the SDM) and a host of non-architectural performance monitoring capabilities. The processor provides two general-purpose performance counters (IA32_PMC0, IA32_PMC1) and three fixed-function performance counters (IA32_FIXED_CTR0, IA32_FIXED_CTR1, IA32_FIXED_CTR2).

Non-architectural performance monitoring in the processor uses the IA32_PERFEVTSELx MSR to configure a set of non-architecture performance monitoring events to be counted by the corresponding general-purpose performance counter. The list of non-architectural performance monitoring events are listed in the Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume 2 : Events document located [here](#).

The bit fields within each IA32_PERFEVTSELx MSR are defined in Figure 18-6 and described in Section 18.2.1.1 and Section 18.2.3 in the SDM. The processor supports AnyThread counting in three architectural performance monitoring events.

For a full description of the processor tile performance monitoring registers, refer to the section 5.1 Core MSRs.

1.2.1 Enhancements of Performance Monitoring in the Intel® Xeon Phi™ processor Tile

The notable enhancements in Knights Landing tile over SLM core include:

- New events. Several more non-architectural performance monitoring events were added. More details will be disclosed in future revision of this document.
- AnyThread support. This facility is limited to following three architectural events - Unhalted Reference Cycles, Unhalted Core Cycles. Fixed counter (IA32_FIXED_CTR0-3) is also supported.
- PEBS-DLA (Precise Event-Based Sampling-Data Linear Address) support. The processor provides memory address in addition to the SLM PEBS record support on select events. The processor PEBS recording format as reported by IA32_PERF_CAPABILITIES [11:8] is 2.
- Off-core response counting facility. This facility in the processor core allows software to count certain transaction responses between the processor tile to sub-systems outside the tile (untile). Counting off-core response requires additional



event qualification configuration facility in conjunction with IA32_PERFEVTSELx. Two off-core response MSRs are provided to use in conjunction with specific event codes that must be specified with IA32_PERFEVTSELx. Knights Landing expands off-core response capability to match the processor until changes.

- Average request latency measurement. The off-core response counting facility can be combined to use two performance counters to count the occurrences and weighted cycles of transaction requests.
- Last Branch Record (LBR) support changes. LBR is not officially part of the performance monitoring architecture and is often documented separately. It is beneficial, however, to note the LBR changes in the processor is in conjunction with performance monitoring changes. The processor supports 8 LBR pairs per thread. The LBR MSR addresses were changed to 0x680-687 (MSR_LASTBRANCH_x_FROM_IP) and 0x6C0-6C7 (MSR_LASTBRANCH_x_TO_IP) to match the Intel® Xeon® processor based servers (they have 16 LBR pairs).

1.2.1.1 Precise Event-Based Sampling

The processor supports precise event based sampling (PEBS). PEBS is supported using IA32_PMC0 (see also Section 17.4.9, "BTS and DS Save Area" in the SDM).

PEBS uses a debug store mechanism to store a set of architectural state information for the processor. The information provides architectural state of the instruction executed after the instruction that caused the event (See section 18.4.4 in the SDM).

The list of PEBS events supported in the processor is shown in the following table.
[Table1: PEBS Performance Events for the Knights Landing Processor]

Table 1. PEBS Performance Events for the processor

Event Name	Event Select	Sub-event	UMask	Data Linear Address Support
BR_INST_RETIRED	C4H	ALL_BRANCHES	00H	No
		JCC	7EH	No
		TAKEN_JCC	FEH	No
		CALL	F9H	No
		REL_CALL	FDH	No
		IND_CALL	FBH	No
		NON_RETURN_IND	EBH	No
		FAR_BRANCH	BFH	No
		RETURN	F7H	No
BR_MISP_RETIRED	C5H	ALL_BRANCHES	00H	No
		JCC	7EH	No
		TAKEN_JCC	FEH	No
		IND_CALL	FBH	No
		NON_RETURN_IND	EBH	No
		RETURN	F7H	No
continued...				



Event Name	Event Select	Sub-event	UMask	Data Linear Address Support
MEM_UOPS_RETIRE	04H	L2_HIT_LOADS	02H	Yes
		L2_MISS_LOADS	04H	Yes
		DLTB_MISS_LOADS	08H	Yes
RECYCLEQ	03H	LD_BLOCK_ST_FORWARD	01H	Yes
		LD_SPLITS	08H	Yes (split high address is captured)

The PEBS record format supported by processors based on the Intel Silvermont microarchitecture is shown in the following table. Each field in the PEBS record is 64 bits long. [Table 2: PEBS record format for the Knights Landing Processor]

Table 2. PEBS Record Format for the processor

Byte Offset	Field	Byte Offset	Field
00H	R/EFLAGS	60H	R10
08H	R/EIP	68H	R11
10H	R/EAX	70H	R12
18H	R/EBX	78H	R13
20H	R/ECX	80H	R14
28H	R/EDX	88H	R15
30H	R/ESI	90H	IA32_PERF_GLOBAL_STATUS
38H	R/EDI	98H	Data Linear Address (new in the Knights Landing Processor)
40H	R/EBP	A0H	Reserved
48H	R/ESP	A8H	Reserved
50H	R8	B0H	Eventing RIP
58H	R9	B8H	Reserved

1.2.1.2 Offcore Response Event

Event number 0B7H support offcore response monitoring using an associated configuration MSR, MSR_OFFCORE_RSP0 (address 1A6H) in conjunction with umask value 01H or MSR_OFFCORE_RSP1 (address 1A7H) in conjunction with umask value 02H. Table 19-20 lists the event code, mask value and additional off-core configuration MSR that must be programmed to count off-core response events using IA32_PMCx.

Table 3. OffCore Response Event Encoding

Counter	Event Code	UMask	Required Off-core Response MSR
PMC0-1	B7H	01H	MSR_OFFCORE_RSP0 (address 1A6H)
PMC0-1	B7H	02H	MSR_OFFCORE_RSP1 (address 1A7H)



Some of the MSR_OFFCORE_RESP {0,1} register bits are not valid in this processor and their use is reserved. The latest layout of MSR_OFFCORE_RSP0 and MSR_OFFCORE_RSP1 registers have been defined in *Table 1-1. Bit fields of the MSR_OFFCORE_RESP {0, 1} Registers* of the Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume 2 : Events document located [here](#). Bits 15:0 specifies the request type of a transaction request to the uncore. Bits 30:16 specifies supplier information, bits 37:31 specifies snoop response information.

Additionally, MSR_OFFCORE_RSP0 provides bit 38 to enable measurement of average latency of specific type of offcore transaction requests.

1.2.1.3 Average Offcore Request Latency Measurement

Measurement of average latency of offcore transaction requests can be enabled using MSR_OFFCORE_RSP0.[bit 38] with the choice of request type specified in MSR_OFFCORE_RSP0.[bit 15:0].

When average latency measurement is enabled, IA32_PMC0 will accumulate weighted cycles of outstanding transaction requests for the specified transaction request type. At the same time, IA32_PMC1 should be configured to accumulate the number of occurrences each time a new transaction request of specified type is made.

1.3 Performance Monitoring in the Intel® Xeon Phi™ processor Untile

The processor untile performance monitoring facilities are organized into distributed per-component performance monitoring (or 'PMON') units. A PMON unit within a untile component may contain a set of counter registers which can be programmed to count various events within that component. With the exception of the UBox, each PMON unit provides a unit-level control register to synchronize actions across the counters within the box (e.g. to start/stop counting). Events can be collected by reading a set of local counter registers. Each counter register is paired with a dedicated control register used to specify what to count (i.e. through the event select/umask fields) and how to count it.

Each of these boxes communicates with the UBOX which contains registers to control all untile PMU activities. The overflow signals sent back to UBOX from PMON units go through message channel. UBOX sends out freeze signal to all other PMON units using a dedicated wire.

Table 4. The processor Per-Component Performance Monitoring Capabilities

Component	# Instances	#PMON units/ instance	#Counters/ PMON (fixed counter)	Bit Width
Ubox	1	1	2 (1)	48
CHA	38	1	4	48
EDC	8	1(Uclk)+1(Eclk)	4 (2)	48
MC	2	1(Uclk)+3(Dclk)	4 (2)	48
M2PCIE	1	1	4	48
PCU	1	1	4 (2)	48



In both EDC and MC, there are one PMON block in the Uclk domain and one PMON block per channel in Eclk/Dclk domain. The PMON block in the Uclk also covers the events originated from Common Mesh Stop (CMS).

EDC, MC and M2PCIE performance monitoring-related registers reside in the PCIe configuration space. In other words, their address is specified in the Bus:Device:Function:Offset format.

UBOX, CHA and PCU performance monitoring-related registers are accessed via the MSR interface. In other words, they have an MSR address.

1.3.1 Untile Performance Monitoring Registers

The following figure describes the general format of performance monitoring-related registers in the processor untile.

Figure 3. Untile PMON module unit control register format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd														Over	Frz	En	Rsvd						Frz	Rsvd				RstC	RstCt		

Each PMON has one unit level control register. If bit 0 is set to 1, it resets all configuration registers in the PMON module. Bit 1, when set to 1, resets all counter registers to zero. Setting bit 8 to 1 freezes all the counters. Bit 16 and 17 enable freeze and overflow, respectively. Note that the UBOX does not have unit control register.

Figure 4. Untile PMON module unit status register format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd																												Ov3	Ov2	Ov1	Ov0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd																												Ov1	Ov0		

The status register within each PMON module has equal number of LSB bits corresponding the number of counters that indicate the counter overflow when set. Software has to write 1 to clean it when necessary. With the exception of the UBOX, which has two counters, all other untile units have four counters.

Each counter inside PMON module has an associated counter control register defined as the following.

Figure 5. Counter Control register that pairs with each counter

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Threshold								Inve	En	Rsvd	Over	Rsvd	ED	Rst	Rsvd	Umask								Event							

For a full description of all the processor untile performance monitoring registers, please refer to the ensuing chapters.

1.4 Terminology

The table below contains important terms used in this specification. For ease-of-use, numeric entries are listed first, with alpha entries following (for example, "AGP 4x"). Acronyms are then entered in their respective place, with non-acronyms following.



Table 5. Glossary

Word/ Acronym	Definition
#	A “#” symbol after a signal name refers to an active low signal, indicating a signal is in the asserted state when driven to a low level.
ASPM	Active State Power Management
BMC	Baseboard Management Controller
CHA	The functional module that includes the CA (Caching Agent) and HA (Home Agent).
CMP	Core Multi-Processing refers to a single physical package that utilizes multiple cores for multi-processing capabilities.
CPU-Only Reset	Any assertion of the CPU RESET# input signal that does not also assert the PCI RESET signal. The PWRGOOD# signal is not toggled. This capability is not supported by Knights Landing processors.
CSR	Control and Status Registers
DDR4	Fourth generation Double Data Rate synchronous dynamic random access memory (SDRAM) technology.
DIMM	Dual-in-Line Memory Module. A packaging arrangement of memory devices on a socketable substrate.
DMA	Direct Memory Access
DRAM Page (Row)	The DRAM cells selected by the Row Address.
TLB	Translation Look-Aside Buffer. Part of the processor core architecture.
DTS	Digital Thermal Sensor
ECC	Error Correcting Code
GPO	General Purpose Output
Host	This term is used synonymously with processor.
Integrated Memory Controller (IMC)	A memory controller that is integrated in the processor silicon.
I/O	1. Input/Output. 2. When used as a qualifier to a transaction type, specifies that transaction targets Intel® architecture-specific I/O space. (for example, I/O read)
IIO	Integrated I/O Controller in the processor die providing I/O components.
Inband	Communication that is multiplexed on the standard lines of an interface, rather than requiring a dedicated signal.
Intel® 64	Intel® 64 Instruction Set Architecture. The instruction set architecture and programming environment of Intel’s 64-bit processors which is a superset of and compatible with IA-32.
Intel® QuickData Technology	Intel® QuickData Technology is a platform solution designed to maximize the throughput of server data traffic across a broader range of configurations and server environments to achieve faster, scalable, and more reliable I/O.
Intel® Virtualization Technology (Intel® VT)	Processor Virtualization which when used in conjunction with Virtual Machine Monitor software enables multiple, robust independent software environments inside a single platform.
continued...	



Word/ Acronym	Definition
Integrated Heat Spreader (IHS)	A component of the processor package used to enhance the thermal performance of the package. Component thermal solutions interface with the processor at the IHS surface.
Legacy	Functional requirements handed down from previous chipsets or PC compatibility requirements from the past.
Link Layer	The layer of an interface that handles flow control and often error correction by retry.
LRDIMM	Load Reduced Dual In-Line Memory Module
LRU	Least Recently Used. A term used in conjunction with cache allocation policy.
MCDRAM	Multi Channel DRAM. A high band width on package DRAM memory
MMIO	Memory Mapped I/O. Any memory access to PCI Express*.
MMCFG	Memory Mapped Configuration. A memory transaction that accesses configuration space.
Multi-Core Processor	A physical package that contains more than one processor core.
MSR	Model Specific Register as the name implies is model specific and may change from processor model number (n) to processor model number (n+1). An MSR is accessed by setting ECX to the register number and executing either the RDMSR or WRMSR instruction. The RDMSR instruction will place the 64 bits of the MSR in the EDX:EAX register pair. The WRMSR writes the contents of the EDX:EAX register pair into the MSR.
PCH	Platform Controller Hub
PCU	Power Control Unit
PECI	Platform Environment Control Interface
Physical Processor	A package which contains 1 or more cores that share a common connection to the system bus.
Power-On Reset	Also known as Cold Reset - occurs the first time the platform asserts PWRGOOD and RESET_N to the processor.
RAPL	Running Average Power Limit is an interface that defines an average power constraint for a given domain and allows the platform to specify its power information via multiple sources (registers and PECI 3.0). The RAPL interface supports multiple usage models for performance and power management features, such as Intel® Turbo Boost Technology.
RASUM	Reliability, Availability, Serviceability, Usability, and Manageability, which are all important characteristics of servers.
Rank	A group of DRAM chips that fill out the data bus width of the system and are accessed in parallel by each DRAM command.
Ring	"Ring" refers to the interface between the processor core and rest of the uncore components.
RDIMM	Registered Dual In-line Memory Module
SKU	Stock Keeping Unit is a subset of a processor type with specific features, electrical, power and thermal specifications. Not all features are supported on all SKUs. A SKU is based on specific use condition assumption.
SDRAM	Synchronous Dynamic Random Access Memory
SEC/DED	Single-bit Error Correct / Double-symbol Error Detect
SMBus	System Management Bus. Mastered by a system management controller to read and write configuration registers. Signaling and protocol are loosely based on I2C, limited to 100 KHz.
continued...	



Word/ Acronym	Definition
Snooping	A means of ensuring cache coherency by monitoring all coherent accesses on a common multi-drop bus to determine if an access is to information resident within a cache. The Cache Agent ensures coherency by initiating snoops on the processor busses with the address of any line that might appear in a cache on that bus.
Socket	The Knights Landing processor (core + uncore).
SVID	Serial Voltage Identification is a binary pattern output from the processor that tells the voltage regulator the voltage required to operate the processor.
TCC	Thermal Control Circuit is a feature of the processor that is used to cool the processor should the processor temperature exceed a predetermined temperature.
TDP	Thermal Design Power
Tile	A unit in Knights Landing Processor that contains 2 cores, L2 cache, VPU and CHA.
Thread	A Logical Processor
Uncore	The portion of the processor comprising the IMC, IIO and related components.
UP	Uni-processor
Warm Reset	Assertion and deassertion of the RESET_N input signal of the CPU where power is maintained to the processor. Does not include "CPU-Only Reset", or "Poweron reset".

1.5 Register Terminology

The bits in configuration register descriptions will have an assigned attribute from the following table. Bits without a Sticky attribute are set to their default value by a hard reset.

Note: The table below is a comprehensive list of all possible attributes.

Table 6. Register Attribute Definitions

Attribute	Description
RO	Read Only: These bits can only be read by software, writes have no effect. The value of the bits is determined by the hardware only.
RW	Read / Write: These bits can be read and written by software.
RC	Read Clear Variant: These bits can be read by software, and the act of reading them automatically clears them. HW is responsible for writing these bits, and therefore the -V modifier is implied.
W1S	Write 1 to Set: Writing a 1 to these bits will set them to 1. Writing 0 will have no effect. Reading will return indeterminate values.
WO	Write Only: These bits can only be written by microcode, reads return indeterminate values. Microcode that wants to ensure this bit was written must read wherever the side-effect takes place.
RW-O	Read / Write Once: These bits can be read by software. After reset, these bits can only be written by software once, after which the bits becomes 'Read Only'.
RW-L	Read / Write Lock: These bits can be read and written by software. The bits can be made to be 'Read Only' via a separate configuration bit or other logic.
RW-KL	Read / Write Lock: These bits can be read and written by software. The bits can be made to be 'Read Only' via a separate configuration bit or other logic. Fields with this attribute also act as the locking agent for other fields.
continued...	



Attribute	Description
RW1C	Read / Write 1 to Clear: These bits can be read and cleared by software. Writing a '1' to a bit clears it, while writing a '0' to a bit has no effect.
RW0C	Read / Write 0 to Clear: These bits can be read and cleared by software. Writing a '0' to a bit clears it while writing a '1' has no effect.
ROS	RO Sticky: These bits can only be read by software, writes have no effect. The value of the bits is determined by the hardware only. These bits are only re-initialized to their default value by a PWRGOOD reset.
RW1S	Read, Write 1 to Set: These bits can be read. Writing a 1 to a given bit will set it to 1. Writing a 0 to a given bit will have no effect. It is not possible for software to set a bit to "0". The 1->0 transition can only be performed by hardware. These registers are implicitly - V.
RWS	R / W Sticky: These bits can be read and written by software. These bits are only re-initialized to their default value by a PWRGOOD reset.
RW1CS	R / W1C Sticky: These bits can be read and cleared by software. Writing a '1' to a bit clears it, while writing a '0' to a bit has no effect. These bits are only re-initialized to their default value by a PWRGOOD reset.
RW-LB	Read / Write Lock Bypass: Similar to RWL, these bits can be read and written by software. HW can make these bits "Read Only" via a separate configuration bit or other logic. However, RW-LB is a special case where the locking is controlled by the lock-bypass capability that is controlled by the lock-bypass enable bits. Each lock-bypass enable bit enables a set of config request sources that can bypass the lock. The requests sourced from the corresponding bypass enable bits will be lock-bypassed (i.e. RW) while requests sourced from other sources are under lock control (RO). The lock bit and bypass enable bit are generally defined with RWO attributes. Sticky can be used with this attribute (RW-SWB). These bits are only reinitialized to their default values after PWRGOOD. Note that the lock bits may not be sticky, and it is important that they are written to after reset to guarantee that software will not be able to change their values after a reset.
RO-FW	Read Only Forced Write: These bits are read only from the perspective of the cores.
RWS-O	Read / Write Sticky Once: If a register is both sticky and "once" then the sticky value applies to both the register value and the "once" characteristic. Only a PWRGOOD reset will reset both the value and the "once" so that the register can be written to again.
RW-V / RO-V	Read Write Variant / Read Only Variant: These bits may be modified by hardware. Software cannot expect the values to stay unchanged. This is similar to "volatile" in software land.
RWS-V	Read / Write Software Variant: These bits can be read or written by software and may be modified by hardware. Software cannot expect the values to stay unchanged. These bits are re-initialized to their default values by a PWRGOOD reset.
RWS-L	Read / Write Sticky Lock: If a register is both sticky and locked, then the sticky behavior only applies to the value. The sticky behavior of the lock is determined by the register that controls the lock.
RWS-LV	Read / Write Sticky Lock Variant: These bits can be read or written by software and may be modified by hardware. Software cannot expect the values to stay unchanged. These bits are re-initialized to their default values by a PWRGOOD reset. If a register is both sticky and locked, then the sticky behavior only applies to the value. The sticky behavior of the lock is determined by the register that controls the lock.
SMM-RO	Read Only in SMM: These bits can only be read by software while in SMM. Writes in SMM have no effect. Attempting to read or write these bits outside of SMM will cause a #GP exception to be raised.
R/SMM-W	Read / Write Only in SMM: These bits can be read by software inside or outside of SMM but can only be written by software while in SMM. Attempting to write these bits outside of SMM will cause a #GP exception to be raised.
SMM-RW	Read Only in SMM / Write Only in SMM: These bits can only be read and written by software while in SMM. Attempting to write these bits outside of SMM will cause a #GP exception to be raised.
continued...	



Attribute	Description
SMM-RW1C	Read / Write 1 to Clear in SMM: These bits can be read and cleared by software only while in SMM. Writing a '1' to a bit clears it, while writing a '0' to a bit has no effect.
RSVD-P	Reserved - Protected: These bits are reserved for future expansion and their value must not be modified by software. When writing these bits, software must preserve the value read.
RSVD-Z	Reserved - Don't Care: These bits are reserved for future expansion and modifying their value has no effect. Software does not need to preserve the value read.



2.0 Embedded DRAM Controller (EDC) Registers

The EDC is the high bandwidth near-memory controller for the processor. EDC refers to "Embedded DRAM Controller" (i.e. DRAM that is embedded in the processor package). The technology that is used to implement the embedded DRAM for the processor is MCDRAM (Multi-Chip (Stacked) DRAM). Eight channels of MCDRAM are supported by 8 MCDRAM Controllers (EDC). The EDC's are connected to the other components (clusters) within the processor by the internal mesh interconnect fabric.

This chapter describes in detail about the performance monitoring- related registers in the Embedded DRAM Controller(EDC).

The following table [Table 7: EDC0 Register ID/Name Mapping] contains a mapping of the EDC0 UCLK and EDC0 ECLK register names used in this document corresponding to the legacy Perfmon register names used conventionally on other Intel® processors for convenience of the reader. Please note that the same mapping applies to the rest of EDC1-7 UCLK and EDC1-7 ECLK registers.

Table 7. EDC0 Register ID/Name Mapping

Register ID	Legacy Perfmon ID
UCLK_PMON_CTR0_LOW_REG	EDC0_UCLK_MSR_PMON_CTR0_LOW
UCLK_PMON_CTR0_HIGH_REG	EDC0_UCLK_MSR_PMON_CTR0_HIGH
UCLK_PMON_CTR1_LOW_REG	EDC0_UCLK_MSR_PMON_CTR1_LOW
UCLK_PMON_CTR1_HIGH_REG	EDC0_UCLK_MSR_PMON_CTR1_HIGH
UCLK_PMON_CTR2_LOW_REG	EDC0_UCLK_MSR_PMON_CTR2_LOW
UCLK_PMON_CTR2_HIGH_REG	EDC0_UCLK_MSR_PMON_CTR2_HIGH
UCLK_PMON_CTR3_LOW_REG	EDC0_UCLK_MSR_PMON_CTR3_LOW
UCLK_PMON_CTR3_HIGH_REG	EDC0_UCLK_MSR_PMON_CTR3_HIGH
UCLK_PMON_CTRCTL0_REG	EDC0_UCLK_MSR_PMON_CTL0
UCLK_PMON_CTRCTL1_REG	EDC0_UCLK_MSR_PMON_CTL1
UCLK_PMON_CTRCTL2_REG	EDC0_UCLK_MSR_PMON_CTL2
UCLK_PMON_CTRCTL3_REG	EDC0_UCLK_MSR_PMON_CTL3
UCLK_PMON_UNIT_CTL_REG	EDC0_UCLK_MSR_PMON_BOX_CTL
UCLK_PMON_UNIT_STATUS_REG	EDC0_UCLK_MSR_PMON_BOX_STATUS
UCLK_PMON_TIMESTAMP_LOW_REG	EDC0_UCLK_MSR_PMON_UCLK_FIXED_LOW
UCLK_PMON_TIMESTAMP_HIGH_REG	EDC0_UCLK_MSR_PMON_UCLK_FIXED_HIGH
UCLK_PMON_TIMESTAMP_CTL_REG	EDC0_UCLK_MSR_PMON_UCLK_FIXED_CTL
ECLK_PMON_CTR0_LOW_REG	EDC0_ECLK_MSR_PMON_CTR0_LOW
ECLK_PMON_CTR0_HIGH_REG	EDC0_ECLK_MSR_PMON_CTR0_HIGH
continued...	



Register ID	Legacy Perfmon ID
ECLK_PMON_CTR1_LOW_REG	EDC0_ECLK_MSR_PMON_CTR1_LOW
ECLK_PMON_CTR1_HIGH_REG	EDC0_ECLK_MSR_PMON_CTR1_HIGH
ECLK_PMON_CTR2_LOW_REG	EDC0_ECLK_MSR_PMON_CTR2_LOW
ECLK_PMON_CTR2_HIGH_REG	EDC0_ECLK_MSR_PMON_CTR2_HIGH
ECLK_PMON_CTR3_LOW_REG	EDC0_ECLK_MSR_PMON_CTR3_LOW
ECLK_PMON_CTR3_HIGH_REG	EDC0_ECLK_MSR_PMON_CTR3_HIGH
ECLK_PMON_CTRCTL0_REG	EDC0_ECLK_MSR_PMON_CTL0
ECLK_PMON_CTRCTL1_REG	EDC0_ECLK_MSR_PMON_CTL1
ECLK_PMON_CTRCTL2_REG	EDC0_ECLK_MSR_PMON_CTL2
ECLK_PMON_CTRCTL3_REG	EDC0_ECLK_MSR_PMON_CTL3
ECLK_PMON_UNIT_CTL_REG	EDC0_ECLK_MSR_PMON_BOX_CTL
ECLK_PMON_UNIT_STATUS_REG	EDC0_ECLK_MSR_PMON_BOX_STATUS
ECLK_PMON_TIMESTAMP_LOW_REG	EDC0_ECLK_MSR_PMON_ECLK_FIXED_LOW
ECLK_PMON_TIMESTAMP_HIGH_REG	EDC0_ECLK_MSR_PMON_ECLK_FIXED_HIGH
ECLK_PMON_TIMESTAMP_CTL_REG	EDC0_ECLK_MSR_PMON_ECLK_FIXED_CTL

2.1 Bus: 2, Device: 15, Function: 0 (CFG)

Table 8. Summary of Bus: 2, Device: 15, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 41	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 41	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 41	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 41	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 42	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 42	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 42	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 42	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 42	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 43	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 44	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 45	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 46	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 47	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 47	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 47	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 48	1h

2.1.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 15	Function: 0	Offset: 400
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.	

2.1.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 15	Function: 0	Offset: 404
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.	

2.1.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 15	Function: 0	Offset: 408
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.	

2.1.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 15	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	



2.1.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 15		Function: 0	Offset: 410
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.		

2.1.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 15		Function: 0	Offset: 414
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.		

2.1.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 15		Function: 0	Offset: 418
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.		

2.1.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 15		Function: 0	Offset: 41C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.		

2.1.9 UCLK_PMON_CTRCTLO_REG

This register controls the operation of the Uclk PMON Counter0.



Bus: 2		Device: 15	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.1.10 UCLK_PMON_CTLCTL1_REG

This register controls the operation of the Uclk PMON Counter1.



Bus: 2		Device: 15	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.1.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.



Bus: 2		Device: 15	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.1.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.



Bus: 2		Device: 15	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.1.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 15	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 15	Function: 0	Offset: 430
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrs — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.1.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 15	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.1.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 15	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

2.1.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.



Bus: 2		Device: 15	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

2.1.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 15	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.2 Bus: 2, Device: 16, Function: 0 (CFG)

Table 9. Summary of Bus: 2, Device: 16, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 49	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 49	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 49	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 49	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 49	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 50	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 50	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 50	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 50	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 51	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 52	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 53	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 54	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 54	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 55	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 55	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 55	1h



2.2.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 16	Function: 0	Offset: 400
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.	

2.2.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 16	Function: 0	Offset: 404
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.	

2.2.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 16	Function: 0	Offset: 408
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.	

2.2.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 16	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	

2.2.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 16	Function: 0	Offset: 410
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.	



2.2.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 16		Function: 0	Offset: 414
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.		

2.2.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 16		Function: 0	Offset: 418
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.		

2.2.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 16		Function: 0	Offset: 41C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.		

2.2.9 UCLK_PMON_CTRCTLO_REG

This register controls the operation of the Uclk PMON Counter0.

Bus: 2		Device: 16		Function: 0		Offset: 420	
Bit	Attr	Default	Description				
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.				
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.				
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.				
21	RO	0h	Reserved (RSVD) — Reserved.				
							<i>continued...</i>



Bus: 2		Device: 16	Function: 0	Offset: 420
Bit	Attr	Default	Description	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.2.10 UCLK_PMON_CTL1_REG

This register controls the operation of the Uclk PMON Counter1.

Bus: 2		Device: 16	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 16	Function: 0	Offset: 424
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.2.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.

Bus: 2		Device: 16	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tilde. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
continued...				

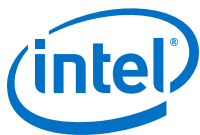


Bus: 2		Device: 16	Function: 0	Offset: 428
Bit	Attr	Default	Description	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.2.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.

Bus: 2		Device: 16	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 16	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.2.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 16	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrls — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.2.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 16	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.2.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 16		Function: 0	Offset: 44C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.		

2.2.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 16		Function: 0	Offset: 450
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.		

2.2.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 16		Function: 0	Offset: 454
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		

2.3 Bus: 2, Device: 17, Function: 0 (CFG)

Table 10. Summary of Bus: 2, Device: 17, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 56	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 56	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 56	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 56	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 57	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 57	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 57	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 57	0h
420–423h	4	UCLK_PMON_CTRCTLO_REG on page 57	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
424–427h	4	UCLK_PMON_CTLCTL1_REG on page 58	0h
428–42Bh	4	UCLK_PMON_CTLCTL2_REG on page 59	0h
42C–42Fh	4	UCLK_PMON_CTLCTL3_REG on page 60	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 61	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 62	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 62	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 62	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 63	1h

2.3.1 UCLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 17		Function: 0		Offset: 400	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.				

2.3.2 UCLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 17		Function: 0		Offset: 404	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.				

2.3.3 UCLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 17		Function: 0		Offset: 408	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.				

2.3.4 UCLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 17	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	

2.3.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 17	Function: 0	Offset: 410
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.	

2.3.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 17	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

2.3.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 17	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

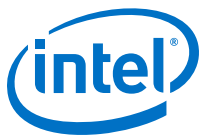
2.3.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 17	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

2.3.9 UCLK_PMON_CTRCTL0_REG

This register controls the operation of the Uclk PMON Counter0.



Bus: 2		Device: 17	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.3.10 UCLK_PMON_CTRCTL1_REG

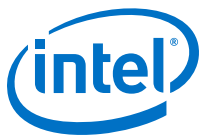
This register controls the operation of the Uclk PMON Counter1.



Bus: 2		Device: 17	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.3.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.



Bus: 2		Device: 17	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.3.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.



Bus: 2		Device: 17	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.3.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 17	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 17	Function: 0	Offset: 430
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctr — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.3.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 17	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.3.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 17	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

2.3.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.



Bus: 2		Device: 17	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

2.3.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 17	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.4 Bus: 2, Device: 18, Function: 0 (CFG)

Table 11. Summary of Bus: 2, Device: 18, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 64	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 64	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 64	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 64	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 64	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 65	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 65	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 65	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 65	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 66	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 67	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 68	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 69	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 69	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 70	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 70	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 70	1h



2.4.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 18		Function: 0		Offset: 400	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.				

2.4.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 18		Function: 0		Offset: 404	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.				

2.4.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 18		Function: 0		Offset: 408	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.				

2.4.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 18		Function: 0		Offset: 40C	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.				

2.4.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 18		Function: 0		Offset: 410	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.				



2.4.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 18	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

2.4.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 18	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

2.4.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 18	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

2.4.9 UCLK_PMON_CTRCTL0_REG

This register controls the operation of the Uclk PMON Counter0.

Bus: 2		Device: 18	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
continued...				



Bus: 2		Device: 18	Function: 0	Offset: 420
Bit	Attr	Default	Description	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.4.10 UCLK_PMON_CTL1_REG

This register controls the operation of the Uclk PMON Counter1.

Bus: 2		Device: 18	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 18	Function: 0	Offset: 424
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.4.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.

Bus: 2		Device: 18	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 18	Function: 0	Offset: 428
Bit	Attr	Default	Description	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.4.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.

Bus: 2		Device: 18	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 18	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.4.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 18	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrs — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.4.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 18	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.4.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 18		Function: 0	Offset: 44C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.		

2.4.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 18		Function: 0	Offset: 450
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.		

2.4.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 18		Function: 0	Offset: 454
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		

2.5 Bus: 2, Device: 19, Function: 0 (CFG)

Table 12. Summary of Bus: 2, Device: 19, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 71	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 71	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 71	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 71	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 72	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 72	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 72	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 72	0h
420–423h	4	UCLK_PMON_CTRCTLO_REG on page 72	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
424–427h	4	UCLK_PMON_CTL1_REG on page 73	0h
428–42Bh	4	UCLK_PMON_CTL2_REG on page 74	0h
42C–42Fh	4	UCLK_PMON_CTL3_REG on page 75	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 76	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 77	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 77	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 77	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 78	1h

2.5.1 UCLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 19		Function: 0		Offset: 400	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.				

2.5.2 UCLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 19		Function: 0		Offset: 404	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.				

2.5.3 UCLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 19		Function: 0		Offset: 408	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.				

2.5.4 UCLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 19	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	

2.5.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 19	Function: 0	Offset: 410
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.	

2.5.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 19	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

2.5.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 19	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

2.5.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 19	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

2.5.9 UCLK_PMON_CTRCTLO_REG

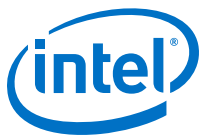
This register controls the operation of the Uclk PMON Counter0.



Bus: 2		Device: 19	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.5.10 UCLK_PMON_CTLCTL1_REG

This register controls the operation of the Uclk PMON Counter1.



Bus: 2		Device: 19	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.5.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.



Bus: 2		Device: 19	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.5.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.



Bus: 2		Device: 19	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.5.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 19	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 19	Function: 0	Offset: 430
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctr — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.5.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 19	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.5.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 19	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

2.5.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.



Bus: 2		Device: 19	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

2.5.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 19	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.6 Bus: 2, Device: 20, Function: 0 (CFG)

Table 13. Summary of Bus: 2, Device: 20, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 79	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 79	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 79	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 79	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 79	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 80	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 80	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 80	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 80	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 81	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 82	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 83	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 84	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 84	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 85	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 85	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 85	1h



2.6.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 20	Function: 0	Offset: 400
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.	

2.6.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 20	Function: 0	Offset: 404
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.	

2.6.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 20	Function: 0	Offset: 408
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.	

2.6.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 20	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	

2.6.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 20	Function: 0	Offset: 410
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.	



2.6.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 20		Function: 0	Offset: 414
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.		

2.6.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 20		Function: 0	Offset: 418
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.		

2.6.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 20		Function: 0	Offset: 41C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.		

2.6.9 UCLK_PMON_CTRCTLO_REG

This register controls the operation of the Uclk PMON Counter0.

Bus: 2		Device: 20		Function: 0	Offset: 420
Bit	Attr	Default	Description		
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.		
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.		
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.		
21	RO	0h	Reserved (RSVD) — Reserved.		
continued...					



Bus: 2		Device: 20	Function: 0	Offset: 420
Bit	Attr	Default	Description	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.6.10 UCLK_PMON_CTL1_REG

This register controls the operation of the Uclk PMON Counter1.

Bus: 2		Device: 20	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 20	Function: 0	Offset: 424
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uklk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uklk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uklk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uklk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.6.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.

Bus: 2		Device: 20	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uklk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uklk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uklk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uklk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...

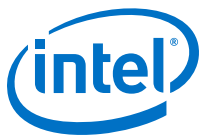


Bus: 2		Device: 20	Function: 0	Offset: 428
Bit	Attr	Default	Description	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.6.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.

Bus: 2		Device: 20	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 20	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.6.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 20	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrs — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.6.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 20	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.6.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 20	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

2.6.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 20	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

2.6.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 20	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.7 Bus: 2, Device: 21, Function: 0 (CFG)

Table 14. Summary of Bus: 2, Device: 21, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 86	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 86	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 86	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 86	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 87	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 87	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 87	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 87	0h
420–423h	4	UCLK_PMON_CTRCTLO_REG on page 87	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
424–427h	4	UCLK_PMON_CTLCTL1_REG on page 88	0h
428–42Bh	4	UCLK_PMON_CTLCTL2_REG on page 89	0h
42C–42Fh	4	UCLK_PMON_CTLCTL3_REG on page 90	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 91	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 92	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 92	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 92	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 93	1h

2.7.1 UCLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 21		Function: 0		Offset: 400	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.				

2.7.2 UCLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 21		Function: 0		Offset: 404	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.				

2.7.3 UCLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 21		Function: 0		Offset: 408	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.				

2.7.4 UCLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 21	Function: 0	Offset: 40C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.	

2.7.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 21	Function: 0	Offset: 410
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.	

2.7.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 21	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

2.7.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 21	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

2.7.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 21	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

2.7.9 UCLK_PMON_CTRCTL0_REG

This register controls the operation of the Uclk PMON Counter0.



Bus: 2		Device: 21	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.7.10 UCLK_PMON_CTLCTL1_REG

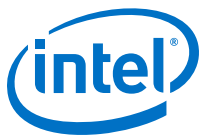
This register controls the operation of the Uclk PMON Counter1.



Bus: 2		Device: 21	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.7.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.



Bus: 2		Device: 21	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.7.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.



Bus: 2		Device: 21	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.7.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 21	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 21	Function: 0	Offset: 430
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uklk_pmon_boxctl_rstctr s — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uklk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.7.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 21	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.7.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 21	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uklk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

2.7.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.



Bus: 2		Device: 21	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

2.7.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 21	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.8 Bus: 2, Device: 22, Function: 0 (CFG)

Table 15. Summary of Bus: 2, Device: 22, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 94	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 94	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 94	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 94	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 94	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 95	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 95	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 95	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 95	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 96	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 97	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 98	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 99	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 99	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 100	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 100	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 100	1h



2.8.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 22		Function: 0		Offset: 400	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uklk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.				

2.8.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 22		Function: 0		Offset: 404	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uklk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.				

2.8.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 22		Function: 0		Offset: 408	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uklk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.				

2.8.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 22		Function: 0		Offset: 40C	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uklk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.				

2.8.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 22		Function: 0		Offset: 410	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uklk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.				



2.8.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 22	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

2.8.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 22	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

2.8.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 22	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

2.8.9 UCLK_PMON_CTRCTLO_REG

This register controls the operation of the Uclk PMON Counter0.

Bus: 2		Device: 22	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
continued...				



Bus: 2		Device: 22	Function: 0	Offset: 420
Bit	Attr	Default	Description	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

2.8.10 UCLK_PMON_CTL1_REG

This register controls the operation of the Uclk PMON Counter1.

Bus: 2		Device: 22	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



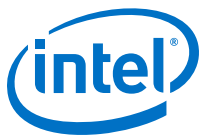
Bus: 2		Device: 22	Function: 0	Offset: 424
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

2.8.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.

Bus: 2		Device: 22	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 22	Function: 0	Offset: 428
Bit	Attr	Default	Description	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

2.8.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.

Bus: 2		Device: 22	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 22	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

2.8.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Uclk PMON.

Bus: 2		Device: 22	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrs — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.8.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Uclk PMON.

Bus: 2		Device: 22	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.8.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 22		Function: 0	Offset: 44C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.		

2.8.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 22		Function: 0	Offset: 450
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.		

2.8.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 22		Function: 0	Offset: 454
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		

2.9 Bus: 2, Device: 24, Function: 2 (CFG)

Table 16. Summary of Bus: 2, Device: 24, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 101	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 101	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 101	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 101	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 102	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 102	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 102	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 102	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 102	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A24–A27h	4	ECLK_PMON_CTL1_REG on page 103	0h
A28–A2Bh	4	ECLK_PMON_CTL2_REG on page 104	0h
A2C–A2Fh	4	ECLK_PMON_CTL3_REG on page 105	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 106	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 107	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 107	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 107	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 108	1h

2.9.1 ECLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 24		Function: 2	Offset: A00
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctl0_low — Eclk PMON Counter0 low 32 bits.		

2.9.2 ECLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 24		Function: 2	Offset: A04
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctl0_high — Eclk PMON Counter0 high 16 bits.		

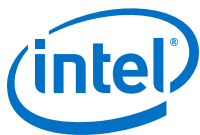
2.9.3 ECLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 24		Function: 2	Offset: A08
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctl1_low — Eclk PMON Counter1 low 32 bits.		

2.9.4 ECLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 24	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.9.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 24	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	

2.9.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 24	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.9.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 24	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.9.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 24	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.9.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.



Bus: 2		Device: 24	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.9.10 ECLK_PMON_CTLCTL1_REG

This register controls the operation of the Eclk PMON Counter1.



Bus: 2		Device: 24	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.9.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.



Bus: 2		Device: 24	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.9.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.



Bus: 2		Device: 24	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.9.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 24	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 24	Function: 2	Offset: A30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrs — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.9.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 24	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.9.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 24	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.9.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.



Bus: 2		Device: 24	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.9.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 24	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.10 Bus: 2, Device: 25, Function: 2 (CFG)

Table 17. Summary of Bus: 2, Device: 25, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 109	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 109	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 109	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 109	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 109	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 110	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 110	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 110	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 110	0h
A24–A27h	4	ECLK_PMON_CTRCTL1_REG on page 111	0h
A28–A2Bh	4	ECLK_PMON_CTRCTL2_REG on page 112	0h
A2C–A2Fh	4	ECLK_PMON_CTRCTL3_REG on page 113	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 114	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 114	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 115	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 115	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 115	1h



2.10.1 ECLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 25	Function: 2	Offset: A00
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.	

2.10.2 ECLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 25	Function: 2	Offset: A04
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.	

2.10.3 ECLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 25	Function: 2	Offset: A08
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.	

2.10.4 ECLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 25	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.10.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 25	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	



2.10.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 25		Function: 2	Offset: A14
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.		

2.10.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 25		Function: 2	Offset: A18
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.		

2.10.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 25		Function: 2	Offset: A1C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.		

2.10.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.

Bus: 2		Device: 25		Function: 2	Offset: A20
Bit	Attr	Default	Description		
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.		
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.		
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.		
21	RO	0h	Reserved (RSVD) — Reserved.		
continued...					



Bus: 2		Device: 25	Function: 2	Offset: A20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.10.10 ECLK_PMON_CTL1_REG

This register controls the operation of the Eclk PMON Counter1.

Bus: 2		Device: 25	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 25	Function: 2	Offset: A24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.10.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.

Bus: 2		Device: 25	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...

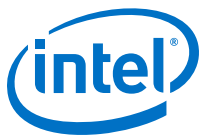


Bus: 2		Device: 25	Function: 2	Offset: A28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.10.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.

Bus: 2		Device: 25	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 25	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.10.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 25	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrs — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.10.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 25	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.10.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 25	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.10.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 25	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.10.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 25	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.11 Bus: 2, Device: 26, Function: 2 (CFG)

Table 18. Summary of Bus: 2, Device: 26, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 116	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 116	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 116	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 116	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 117	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 117	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 117	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 117	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 117	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A24–A27h	4	ECLK_PMON_CTLCTL1_REG on page 118	0h
A28–A2Bh	4	ECLK_PMON_CTLCTL2_REG on page 119	0h
A2C–A2Fh	4	ECLK_PMON_CTLCTL3_REG on page 120	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 121	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 122	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 122	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 122	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 123	1h

2.11.1 ECLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 26		Function: 2		Offset: A00	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.				

2.11.2 ECLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 26		Function: 2		Offset: A04	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.				

2.11.3 ECLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 26		Function: 2		Offset: A08	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.				

2.11.4 ECLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 26	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.11.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 26	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	

2.11.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 26	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.11.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 26	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.11.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 26	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.11.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.



Bus: 2		Device: 26	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.11.10 ECLK_PMON_CTL1_REG

This register controls the operation of the Eclk PMON Counter1.



Bus: 2		Device: 26	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.11.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.



Bus: 2		Device: 26	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.11.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.



Bus: 2		Device: 26	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.11.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 26	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 26	Function: 2	Offset: A30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctr — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.11.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 26	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.11.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 26	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.11.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.



Bus: 2		Device: 26	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.11.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 26	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.12 Bus: 2, Device: 27, Function: 2 (CFG)

Table 19. Summary of Bus: 2, Device: 27, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 124	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 124	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 124	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 124	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 124	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 125	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 125	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 125	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 125	0h
A24–A27h	4	ECLK_PMON_CTRCTL1_REG on page 126	0h
A28–A2Bh	4	ECLK_PMON_CTRCTL2_REG on page 127	0h
A2C–A2Fh	4	ECLK_PMON_CTRCTL3_REG on page 128	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 129	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 129	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 130	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 130	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 130	1h



2.12.1 ECLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 27		Function: 2	Offset: A00
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.		

2.12.2 ECLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 27		Function: 2	Offset: A04
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.		

2.12.3 ECLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 27		Function: 2	Offset: A08
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.		

2.12.4 ECLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 27		Function: 2	Offset: A0C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.		

2.12.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 27		Function: 2	Offset: A10
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.		



2.12.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 27	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.12.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 27	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.12.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 27	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.12.9 ECLK_PMON_CTRCTL0_REG

This register controls the operation of the Eclk PMON Counter0.

Bus: 2		Device: 27	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 27	Function: 2	Offset: A20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.12.10 ECLK_PMON_CTLCTL1_REG

This register controls the operation of the Eclk PMON Counter1.

Bus: 2		Device: 27	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 27	Function: 2	Offset: A24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.12.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.

Bus: 2		Device: 27	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 27	Function: 2	Offset: A28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.12.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.

Bus: 2		Device: 27	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 27	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.12.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 27	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrs — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.12.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 27	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.12.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 27	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.12.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 27	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.12.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 27	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.13 Bus: 2, Device: 28, Function: 2 (CFG)

Table 20. Summary of Bus: 2, Device: 28, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 131	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 131	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 131	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 131	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 132	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 132	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 132	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 132	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 132	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A24–A27h	4	ECLK_PMON_CTL1_REG on page 133	0h
A28–A2Bh	4	ECLK_PMON_CTL2_REG on page 134	0h
A2C–A2Fh	4	ECLK_PMON_CTL3_REG on page 135	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 136	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 137	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 137	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 137	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 138	1h

2.13.1 ECLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 28		Function: 2	Offset: A00
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctl0_low — Eclk PMON Counter0 low 32 bits.		

2.13.2 ECLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 28		Function: 2	Offset: A04
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctl0_high — Eclk PMON Counter0 high 16 bits.		

2.13.3 ECLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 28		Function: 2	Offset: A08
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctl1_low — Eclk PMON Counter1 low 32 bits.		

2.13.4 ECLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 28	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.13.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 28	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	

2.13.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 28	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.13.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 28	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.13.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 28	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.13.9 ECLK_PMON_CTRCTLO_REG

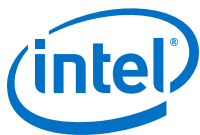
This register controls the operation of the Eclk PMON Counter0.



Bus: 2		Device: 28	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.13.10 ECLK_PMON_CTLCTL1_REG

This register controls the operation of the Eclk PMON Counter1.



Bus: 2		Device: 28	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.13.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.



Bus: 2		Device: 28	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.13.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.



Bus: 2		Device: 28	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.13.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 28	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 28	Function: 2	Offset: A30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrs — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.13.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 28	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.13.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 28	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.13.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.



Bus: 2		Device: 28	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.13.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 28	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.14 Bus: 2, Device: 29, Function: 2 (CFG)

Table 21. Summary of Bus: 2, Device: 29, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 139	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 139	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 139	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 139	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 139	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 140	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 140	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 140	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 140	0h
A24–A27h	4	ECLK_PMON_CTRCTL1_REG on page 141	0h
A28–A2Bh	4	ECLK_PMON_CTRCTL2_REG on page 142	0h
A2C–A2Fh	4	ECLK_PMON_CTRCTL3_REG on page 143	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 144	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 144	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 145	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 145	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 145	1h



2.14.1 ECLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 29	Function: 2	Offset: A00
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.	

2.14.2 ECLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 29	Function: 2	Offset: A04
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.	

2.14.3 ECLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 29	Function: 2	Offset: A08
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.	

2.14.4 ECLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 29	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.14.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 29	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	



2.14.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 29		Function: 2	Offset: A14
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.		

2.14.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 29		Function: 2	Offset: A18
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.		

2.14.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 29		Function: 2	Offset: A1C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.		

2.14.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.

Bus: 2		Device: 29		Function: 2	Offset: A20
Bit	Attr	Default	Description		
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.		
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.		
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.		
21	RO	0h	Reserved (RSVD) — Reserved.		
continued...					



Bus: 2		Device: 29	Function: 2	Offset: A20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.14.10 ECLK_PMON_CTLCTL1_REG

This register controls the operation of the Eclk PMON Counter1.

Bus: 2		Device: 29	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 29	Function: 2	Offset: A24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.14.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.

Bus: 2		Device: 29	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 29	Function: 2	Offset: A28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.14.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.

Bus: 2		Device: 29	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 29	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.14.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 29	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrls — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.14.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 29	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.14.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 29	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.14.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 29	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.14.17 ECLK_PMON_TIMESTAMP_CTL_REG

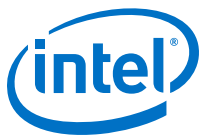
This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 29	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.15 Bus: 2, Device: 30, Function: 2 (CFG)

Table 22. Summary of Bus: 2, Device: 30, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 146	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 146	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 146	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 146	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 147	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 147	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 147	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 147	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 147	0h
<i>continued...</i>			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A24–A27h	4	ECLK_PMON_CTLCTL1_REG on page 148	0h
A28–A2Bh	4	ECLK_PMON_CTLCTL2_REG on page 149	0h
A2C–A2Fh	4	ECLK_PMON_CTLCTL3_REG on page 150	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 151	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 152	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 152	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 152	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 153	1h

2.15.1 ECLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 30		Function: 2		Offset: A00	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.				

2.15.2 ECLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 30		Function: 2		Offset: A04	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.				

2.15.3 ECLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 30		Function: 2		Offset: A08	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.				

2.15.4 ECLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 30	Function: 2	Offset: A0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.	

2.15.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 30	Function: 2	Offset: A10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.	

2.15.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 30	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.15.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 30	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.15.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 30	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.15.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.



Bus: 2		Device: 30	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.15.10 ECLK_PMON_CTL1_REG

This register controls the operation of the Eclk PMON Counter1.



Bus: 2		Device: 30	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.15.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.



Bus: 2		Device: 30	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.15.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.



Bus: 2		Device: 30	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.15.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 30	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 30	Function: 2	Offset: A30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctr — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.15.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 30	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

2.15.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 30	Function: 2	Offset: A3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.	

2.15.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.



Bus: 2		Device: 30	Function: 2	Offset: A40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.	

2.15.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 30	Function: 2	Offset: A44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

2.16 Bus: 2, Device: 31, Function: 2 (CFG)

Table 23. Summary of Bus: 2, Device: 31, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A00–A03h	4	ECLK_PMON_CTR0_LOW_REG on page 154	0h
A04–A07h	4	ECLK_PMON_CTR0_HIGH_REG on page 154	0h
A08–A0Bh	4	ECLK_PMON_CTR1_LOW_REG on page 154	0h
A0C–A0Fh	4	ECLK_PMON_CTR1_HIGH_REG on page 154	0h
A10–A13h	4	ECLK_PMON_CTR2_LOW_REG on page 154	0h
A14–A17h	4	ECLK_PMON_CTR2_HIGH_REG on page 155	0h
A18–A1Bh	4	ECLK_PMON_CTR3_LOW_REG on page 155	0h
A1C–A1Fh	4	ECLK_PMON_CTR3_HIGH_REG on page 155	0h
A20–A23h	4	ECLK_PMON_CTRCTL0_REG on page 155	0h
A24–A27h	4	ECLK_PMON_CTRCTL1_REG on page 156	0h
A28–A2Bh	4	ECLK_PMON_CTRCTL2_REG on page 157	0h
A2C–A2Fh	4	ECLK_PMON_CTRCTL3_REG on page 158	0h
A30–A33h	4	ECLK_PMON_UNIT_CTL_REG on page 159	0h
A34–A37h	4	ECLK_PMON_UNIT_STATUS_REG on page 159	0h
A3C–A3Fh	4	ECLK_PMON_TIMESTAMP_LOW_REG on page 160	0h
A40–A43h	4	ECLK_PMON_TIMESTAMP_HIGH_REG on page 160	0h
A44–A47h	4	ECLK_PMON_TIMESTAMP_CTL_REG on page 160	1h



2.16.1 ECLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 31		Function: 2	Offset: A00
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr0_low — Eclk PMON Counter0 low 32 bits.		

2.16.2 ECLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 31		Function: 2	Offset: A04
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr0_high — Eclk PMON Counter0 high 16 bits.		

2.16.3 ECLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 31		Function: 2	Offset: A08
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr1_low — Eclk PMON Counter1 low 32 bits.		

2.16.4 ECLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 31		Function: 2	Offset: A0C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_ctr1_high — Eclk PMON Counter1 high 16 bits.		

2.16.5 ECLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 31		Function: 2	Offset: A10
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_ctr2_low — Eclk PMON Counter2 low 32 bits.		



2.16.6 ECLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 31	Function: 2	Offset: A14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr2_high — Eclk PMON Counter2 high 16 bits.	

2.16.7 ECLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 31	Function: 2	Offset: A18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_eclk_pmon_ctr3_low — Eclk PMON Counter3 low 32 bits.	

2.16.8 ECLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit EDC ECLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 31	Function: 2	Offset: A1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_eclk_pmon_ctr3_high — Eclk PMON Counter3 high 16 bits.	

2.16.9 ECLK_PMON_CTRCTLO_REG

This register controls the operation of the Eclk PMON Counter0.

Bus: 2		Device: 31	Function: 2	Offset: A20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl0_thresh — Eclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl0_inv — Eclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl0_en — Eclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
continued...				



Bus: 2		Device: 31	Function: 2	Offset: A20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_eclk_pmon_ctl0_oven — Eclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl0_ed — Eclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl0_rst — Eclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl0_umask — Eclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl0_evsel — Eclk PMON Counter0 Event Select. Selects the event to be counted.	

2.16.10 ECLK_PMON_CTLCTL1_REG

This register controls the operation of the Eclk PMON Counter1.

Bus: 2		Device: 31	Function: 2	Offset: A24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl1_thresh — Eclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl1_inv — Eclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl1_en — Eclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl1_oven — Eclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 31	Function: 2	Offset: A24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl1_ed — Eclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl1_rst — Eclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl1_umask — Eclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl1_evsel — Eclk PMON Counter1 Event Select. Selects the event to be counted.	

2.16.11 ECLK_PMON_CTL2_REG

This register controls the operation of the Eclk PMON Counter2.

Bus: 2		Device: 31	Function: 2	Offset: A28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl2_thresh — Eclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl2_inv — Eclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl2_en — Eclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl2_oven — Eclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 31	Function: 2	Offset: A28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_eclk_pmon_ctl2_ed — Eclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl2_rst — Eclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl2_umask — Eclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl2_evsel — Eclk PMON Counter2 Event Select. Selects the event to be counted.	

2.16.12 ECLK_PMON_CTL3_REG

This register controls the operation of the Eclk PMON Counter3.

Bus: 2		Device: 31	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_eclk_pmon_ctl3_thresh — Eclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_eclk_pmon_ctl3_inv — Eclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_eclk_pmon_ctl3_en — Eclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_eclk_pmon_ctl3_oven — Eclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_eclk_pmon_ctl3_ed — Eclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_eclk_pmon_ctl3_rst — Eclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 31	Function: 2	Offset: A2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_eclk_pmon_ctl3_umask — Eclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_eclk_pmon_ctl3_evsel — Eclk PMON Counter3 Event Select. Selects the event to be counted.	

2.16.13 ECLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the EDC Eclk PMON.

Bus: 2		Device: 31	Function: 2	Offset: A30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_eclk_pmon_boxctl_freeze — Eclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_eclk_pmon_boxctl_rstctrs — Eclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_eclk_pmon_boxctl_rstcfg — Eclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

2.16.14 ECLK_PMON_UNIT_STATUS_REG

This register provides unit status of the EDC Eclk PMON.

Bus: 2		Device: 31	Function: 2	Offset: A34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Eclk PMON Config Status3 register. Set if PMON Counter3 overflows Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Eclk PMON Config Status2 register. Set if PMON Counter2 overflows Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Eclk PMON Config Status1 register. Set if PMON Counter1 overflows Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Eclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



2.16.15 ECLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 31		Function: 2	Offset: A3C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_eclk_pmon_timestamp_low — Eclk PMON Timestamp Counter low 32 bits.		

2.16.16 ECLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of eclk cycles since reset was deasserted.

Bus: 2		Device: 31		Function: 2	Offset: A40
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_eclk_pmon_timestamp_high — Eclk PMON Timestamp Counter high 16 bits.		

2.16.17 ECLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Eclk timestamp counter.

Bus: 2		Device: 31		Function: 2	Offset: A44
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		



3.0 Memory Controller (MC) Registers

The processor implements two Memory Controllers on the processor die. Each memory controller is capable of controlling three DDR4 memory channels. The MC design is derived from the EDC (Near-Memory (MCDRAM) controller) and is a sub-set of EDC in functionality. The main difference from EDC is that the physical interface for MC will be DDR4 IOs. The processor MC will interface with the rest of the Untile via the mesh interface (R2Mem -> Ring-to-MC interface). Therefore, the MC agent is broken into three regions: The front-end ring/mesh interface called the "R2Mem", the core "EDC controller" logic, and three individual "DDR channel controllers/schedulers."

This chapter describes in detail about the performance monitoring- related registers in the memory controller.

The following table [Table 24: MC0 Register ID/Name Mapping] contains a mapping of the MC0 UCLK and MC0 CH0 register names used in this document corresponding to the legacy Perfmon register names used conventionally on other Intel® processors for convenience of the reader. Please note that the same mapping applies to the rest of MC0 CH1, MC0 CH2, MC1 UCLK, MC1 CH0, MC1 CH1, MC1 CH2 registers.

Table 24. MC0 Register ID/Name Mapping

Register ID	Legacy Perfmon ID
UCLK_PMON_CTR0_LOW_REG	MC0_UCLK_MSR_PMON_CTR0_LOW
UCLK_PMON_CTR0_HIGH_REG	MC0_UCLK_MSR_PMON_CTR0_HIGH
UCLK_PMON_CTR1_LOW_REG	MC0_UCLK_MSR_PMON_CTR1_LOW
UCLK_PMON_CTR1_HIGH_REG	MC0_UCLK_MSR_PMON_CTR1_HIGH
UCLK_PMON_CTR2_LOW_REG	MC0_UCLK_MSR_PMON_CTR2_LOW
UCLK_PMON_CTR2_HIGH_REG	MC0_UCLK_MSR_PMON_CTR2_HIGH
UCLK_PMON_CTR3_LOW_REG	MC0_UCLK_MSR_PMON_CTR3_LOW
UCLK_PMON_CTR3_HIGH_REG	MC0_UCLK_MSR_PMON_CTR3_HIGH
UCLK_PMON_CTRCTL0_REG	MC0_UCLK_MSR_PMON_CTL0
UCLK_PMON_CTRCTL1_REG	MC0_UCLK_MSR_PMON_CTL1
UCLK_PMON_CTRCTL2_REG	MC0_UCLK_MSR_PMON_CTL2
UCLK_PMON_CTRCTL3_REG	MC0_UCLK_MSR_PMON_CTL3
UCLK_PMON_UNIT_CTL_REG	MC0_UCLK_MSR_PMON_BOX_CTL
UCLK_PMON_UNIT_STATUS_REG	MC0_UCLK_MSR_PMON_BOX_STATUS
UCLK_PMON_TIMESTAMP_LOW_REG	MC0_UCLK_MSR_PMON_UCLK_FIXED_LOW
UCLK_PMON_TIMESTAMP_HIGH_REG	MC0_UCLK_MSR_PMON_UCLK_FIXED_HIGH
UCLK_PMON_TIMESTAMP_CTL_REG	MC0_UCLK_MSR_PMON_UCLK_FIXED_CTL
<i>continued...</i>	



Register ID	Legacy Perfmon ID
DCLK_PMON_CTR0_LOW_REG	MC0_CH0_MSR_PMON_CTR0_LOW
DCLK_PMON_CTR0_HIGH_REG	MC0_CH0_MSR_PMON_CTR0_HIGH
DCLK_PMON_CTR1_LOW_REG	MC0_CH0_MSR_PMON_CTR1_LOW
DCLK_PMON_CTR1_HIGH_REG	MC0_CH0_MSR_PMON_CTR1_HIGH
DCLK_PMON_CTR2_LOW_REG	MC0_CH0_MSR_PMON_CTR2_LOW
DCLK_PMON_CTR2_HIGH_REG	MC0_CH0_MSR_PMON_CTR2_HIGH
DCLK_PMON_CTR3_LOW_REG	MC0_CH0_MSR_PMON_CTR3_LOW
DCLK_PMON_CTR3_HIGH_REG	MC0_CH0_MSR_PMON_CTR3_HIGH
DCLK_PMON_CTRCTL0_REG	MC0_CH0_MSR_PMON_CTL0
DCLK_PMON_CTRCTL1_REG	MC0_CH0_MSR_PMON_CTL1
DCLK_PMON_CTRCTL2_REG	MC0_CH0_MSR_PMON_CTL2
DCLK_PMON_CTRCTL3_REG	MC0_CH0_MSR_PMON_CTL3
DCLK_PMON_UNIT_CTL_REG	MC0_CH0_MSR_PMON_BOX_CTL
DCLK_PMON_UNIT_STATUS_REG	MC0_CH0_MSR_PMON_BOX_STATUS
DCLK_PMON_TIMESTAMP_LOW_REG	MC0_CH0_MSR_PMON_DCLK_FIXED_LOW
DCLK_PMON_TIMESTAMP_HIGH_REG	MC0_CH0_MSR_PMON_DCLK_FIXED_HIGH
DCLK_PMON_TIMESTAMP_CTL_REG	MC0_CH0_MSR_PMON_DCLK_FIXED_CTL

3.1 Bus: 2, Device: 10, Function: 0 (CFG)

Table 25. Summary of Bus: 2, Device: 10, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 163	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 163	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 163	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 163	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 164	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 164	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 164	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 164	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 164	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 165	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 166	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 167	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 168	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 169	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 169	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 169	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 170	1h

3.1.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0	Offset: 400
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.		

3.1.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0	Offset: 404
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.		

3.1.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0	Offset: 408
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.		

3.1.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0	Offset: 40C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.		



3.1.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0		Offset: 410	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.				

3.1.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0		Offset: 414	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.				

3.1.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0		Offset: 418	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.				

3.1.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 10		Function: 0		Offset: 41C	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.				

3.1.9 UCLK_PMON_CTRCTLO_REG

This register controls the operation of the Uclk PMON Counter0.



Bus: 2		Device: 10	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

3.1.10 UCLK_PMON_CTLCTL1_REG

This register controls the operation of the Uclk PMON Counter1.



Bus: 2		Device: 10	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

3.1.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.



Bus: 2		Device: 10	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

3.1.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.



Bus: 2		Device: 10	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

3.1.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Uclk PMON.

Bus: 2		Device: 10	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 10	Function: 0	Offset: 430
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctr — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.1.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Uclk PMON.

Bus: 2		Device: 10	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

3.1.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 10	Function: 0	Offset: 44C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.	

3.1.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.



Bus: 2		Device: 10	Function: 0	Offset: 450
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.	

3.1.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 10	Function: 0	Offset: 454
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.2 Bus: 2, Device: 8, Function: 2 (CFG)

Table 26. Summary of Bus: 2, Device: 8, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 171	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 171	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 171	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 171	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 171	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 172	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 172	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 172	0h
B20-B23h	4	DCLK_PMON_CTRCTL0_REG on page 172	0h
B24-B27h	4	DCLK_PMON_CTRCTL1_REG on page 173	0h
B28-B2Bh	4	DCLK_PMON_CTRCTL2_REG on page 174	0h
B2C-B2Fh	4	DCLK_PMON_CTRCTL3_REG on page 175	0h
B30-B33h	4	DCLK_PMON_UNIT_CTL_REG on page 176	0h
B34-B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 176	0h
B3C-B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 177	0h
B40-B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 177	0h
B44-B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 177	1h



3.2.1 DCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 2	Offset: B00
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.	

3.2.2 DCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 2	Offset: B04
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.	

3.2.3 DCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 2	Offset: B08
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.	

3.2.4 DCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 2	Offset: B0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.	

3.2.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 2	Offset: B10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.	



3.2.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 2	Offset: B14
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.		

3.2.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 2	Offset: B18
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.		

3.2.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 2	Offset: B1C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.		

3.2.9 DCLK_PMON_CTRCTLO_REG

This register controls the operation of the Dclk PMON Counter0.

Bus: 2		Device: 8		Function: 2		Offset: B20	
Bit	Attr	Default	Description				
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.				
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.				
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.				
21	RO	0h	Reserved (RSVD) — Reserved.				
							<i>continued...</i>



Bus: 2		Device: 8	Function: 2	Offset: B20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.2.10 DCLK_PMON_CTL1_REG

This register controls the operation of the Dclk PMON Counter1.

Bus: 2		Device: 8	Function: 2	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	

continued...



Bus: 2		Device: 8	Function: 2	Offset: B24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.2.11 DCLK_PMON_CTL2_REG

This register controls the operation of the Dclk PMON Counter2.

Bus: 2		Device: 8	Function: 2	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 8	Function: 2	Offset: B28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.2.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.

Bus: 2		Device: 8	Function: 2	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 8	Function: 2	Offset: B2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.2.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 2	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctrs — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.2.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 2	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



3.2.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 8	Function: 2	Offset: B3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.	

3.2.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 8	Function: 2	Offset: B40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.	

3.2.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 8	Function: 2	Offset: B44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.3 Bus: 2, Device: 8, Function: 3 (CFG)

Table 27. Summary of Bus: 2, Device: 8, Function: 3 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 178	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 178	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 178	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 178	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 179	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 179	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 179	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 179	0h
B20-B23h	4	DCLK_PMON_CTRCTLO_REG on page 179	0h
<i>continued...</i>			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B24–B27h	4	DCLK_PMON_CTRCTL1_REG on page 180	0h
B28–B2Bh	4	DCLK_PMON_CTRCTL2_REG on page 181	0h
B2C–B2Fh	4	DCLK_PMON_CTRCTL3_REG on page 182	0h
B30–B33h	4	DCLK_PMON_UNIT_CTL_REG on page 183	0h
B34–B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 184	0h
B3C–B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 184	0h
B40–B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 184	0h
B44–B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 185	1h

3.3.1 DCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 3		Offset: B00	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.				

3.3.2 DCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 3		Offset: B04	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.				

3.3.3 DCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 3		Offset: B08	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.				

3.3.4 DCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 8	Function: 3	Offset: B0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.	

3.3.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 3	Offset: B10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.	

3.3.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 3	Offset: B14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.	

3.3.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 3	Offset: B18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.	

3.3.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 3	Offset: B1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.	

3.3.9 DCLK_PMON_CTRCTLO_REG

This register controls the operation of the Dclk PMON Counter0.



Bus: 2		Device: 8	Function: 3	Offset: B20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.3.10 DCLK_PMON_CTL0_REG

This register controls the operation of the Dclk PMON Counter1.



Bus: 2		Device: 8	Function: 3	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.3.11 DCLK_PMON_CTL2_REG

This register controls the operation of the Dclk PMON Counter2.



Bus: 2		Device: 8	Function: 3	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.3.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.



Bus: 2		Device: 8	Function: 3	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.3.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 3	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 8	Function: 3	Offset: B30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctr — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.3.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 3	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

3.3.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 8	Function: 3	Offset: B3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.	

3.3.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.



Bus: 2		Device: 8	Function: 3	Offset: B40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.	

3.3.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 8	Function: 3	Offset: B44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.4 Bus: 2, Device: 8, Function: 4 (CFG)

Table 28. Summary of Bus: 2, Device: 8, Function: 4 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 186	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 186	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 186	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 186	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 186	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 187	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 187	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 187	0h
B20-B23h	4	DCLK_PMON_CTRCTL0_REG on page 187	0h
B24-B27h	4	DCLK_PMON_CTRCTL1_REG on page 188	0h
B28-B2Bh	4	DCLK_PMON_CTRCTL2_REG on page 189	0h
B2C-B2Fh	4	DCLK_PMON_CTRCTL3_REG on page 190	0h
B30-B33h	4	DCLK_PMON_UNIT_CTL_REG on page 191	0h
B34-B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 191	0h
B3C-B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 192	0h
B40-B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 192	0h
B44-B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 192	1h



3.4.1 DCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 4		Offset: B00	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.				

3.4.2 DCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 4		Offset: B04	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.				

3.4.3 DCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 4		Offset: B08	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.				

3.4.4 DCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 4		Offset: B0C	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.				

3.4.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8		Function: 4		Offset: B10	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.				



3.4.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 4	Offset: B14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.	

3.4.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 4	Offset: B18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.	

3.4.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 8	Function: 4	Offset: B1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.	

3.4.9 DCLK_PMON_CTRCTL0_REG

This register controls the operation of the Dclk PMON Counter0.

Bus: 2		Device: 8	Function: 4	Offset: B20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
continued...				



Bus: 2		Device: 8	Function: 4	Offset: B20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.4.10 DCLK_PMON_CTL1_REG

This register controls the operation of the Dclk PMON Counter1.

Bus: 2		Device: 8	Function: 4	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 8	Function: 4	Offset: B24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.4.11 DCLK_PMON_CTL2_REG

This register controls the operation of the Dclk PMON Counter2.

Bus: 2		Device: 8	Function: 4	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 8	Function: 4	Offset: B28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.4.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.

Bus: 2		Device: 8	Function: 4	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 8	Function: 4	Offset: B2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.4.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 4	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctrs — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.4.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 8	Function: 4	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



3.4.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 8		Function: 4	Offset: B3C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.		

3.4.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 8		Function: 4	Offset: B40
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.		

3.4.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 8		Function: 4	Offset: B44
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		

3.5 Bus: 2, Device: 9, Function: 2 (CFG)

Table 29. Summary of Bus: 2, Device: 9, Function: 2 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 193	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 193	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 193	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 193	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 194	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 194	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 194	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 194	0h
B20-B23h	4	DCLK_PMON_CTRCTL0_REG on page 194	0h
<i>continued...</i>			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B24–B27h	4	DCLK_PMON_CTL1_REG on page 195	0h
B28–B2Bh	4	DCLK_PMON_CTL2_REG on page 196	0h
B2C–B2Fh	4	DCLK_PMON_CTL3_REG on page 197	0h
B30–B33h	4	DCLK_PMON_UNIT_CTL_REG on page 198	0h
B34–B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 199	0h
B3C–B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 199	0h
B40–B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 199	0h
B44–B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 200	1h

3.5.1 DCLK_PMON_CTL0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 2	Offset: B00
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.		

3.5.2 DCLK_PMON_CTL0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 2	Offset: B04
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.		

3.5.3 DCLK_PMON_CTL1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 2	Offset: B08
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.		

3.5.4 DCLK_PMON_CTL1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 9	Function: 2	Offset: B0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.	

3.5.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 2	Offset: B10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.	

3.5.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 2	Offset: B14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.	

3.5.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 2	Offset: B18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.	

3.5.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 2	Offset: B1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.	

3.5.9 DCLK_PMON_CTRCTLO_REG

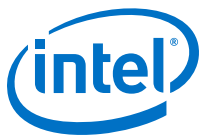
This register controls the operation of the Dclk PMON Counter0.



Bus: 2		Device: 9	Function: 2	Offset: B20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.5.10 DCLK_PMON_CTL1_REG

This register controls the operation of the Dclk PMON Counter1.



Bus: 2		Device: 9	Function: 2	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.5.11 DCLK_PMON_CTLCTL2_REG

This register controls the operation of the Dclk PMON Counter2.



Bus: 2		Device: 9	Function: 2	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.5.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.



Bus: 2		Device: 9	Function: 2	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.5.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 2	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 9	Function: 2	Offset: B30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctr — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.5.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 2	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

3.5.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 9	Function: 2	Offset: B3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.	

3.5.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.



Bus: 2		Device: 9	Function: 2	Offset: B40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.	

3.5.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 9	Function: 2	Offset: B44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.6 Bus: 2, Device: 9, Function: 3 (CFG)

Table 30. Summary of Bus: 2, Device: 9, Function: 3 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 201	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 201	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 201	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 201	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 201	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 202	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 202	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 202	0h
B20-B23h	4	DCLK_PMON_CTRCTL0_REG on page 202	0h
B24-B27h	4	DCLK_PMON_CTRCTL1_REG on page 203	0h
B28-B2Bh	4	DCLK_PMON_CTRCTL2_REG on page 204	0h
B2C-B2Fh	4	DCLK_PMON_CTRCTL3_REG on page 205	0h
B30-B33h	4	DCLK_PMON_UNIT_CTL_REG on page 206	0h
B34-B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 206	0h
B3C-B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 207	0h
B40-B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 207	0h
B44-B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 207	1h



3.6.1 DCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 3	Offset: B00
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.	

3.6.2 DCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 3	Offset: B04
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.	

3.6.3 DCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 3	Offset: B08
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.	

3.6.4 DCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 3	Offset: B0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.	

3.6.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 3	Offset: B10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.	



3.6.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 3	Offset: B14
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.		

3.6.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 3	Offset: B18
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.		

3.6.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 3	Offset: B1C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.		

3.6.9 DCLK_PMON_CTRCTLO_REG

This register controls the operation of the Dclk PMON Counter0.

Bus: 2		Device: 9		Function: 3		Offset: B20	
Bit	Attr	Default	Description				
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.				
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.				
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.				
21	RO	0h	Reserved (RSVD) — Reserved.				
							<i>continued...</i>



Bus: 2		Device: 9	Function: 3	Offset: B20
Bit	Attr	Default	Description	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.6.10 DCLK_PMON_CTL1_REG

This register controls the operation of the Dclk PMON Counter1.

Bus: 2		Device: 9	Function: 3	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 9	Function: 3	Offset: B24
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.6.11 DCLK_PMON_CTL2_REG

This register controls the operation of the Dclk PMON Counter2.

Bus: 2		Device: 9	Function: 3	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 9	Function: 3	Offset: B28
Bit	Attr	Default	Description	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.6.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.

Bus: 2		Device: 9	Function: 3	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued..				



Bus: 2		Device: 9	Function: 3	Offset: B2C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.6.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 3	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctrs — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.6.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 3	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



3.6.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 9	Function: 3	Offset: B3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.	

3.6.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 9	Function: 3	Offset: B40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.	

3.6.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 9	Function: 3	Offset: B44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.7 Bus: 2, Device: 9, Function: 4 (CFG)

Table 31. Summary of Bus: 2, Device: 9, Function: 4 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B00-B03h	4	DCLK_PMON_CTR0_LOW_REG on page 208	0h
B04-B07h	4	DCLK_PMON_CTR0_HIGH_REG on page 208	0h
B08-B0Bh	4	DCLK_PMON_CTR1_LOW_REG on page 208	0h
B0C-B0Fh	4	DCLK_PMON_CTR1_HIGH_REG on page 208	0h
B10-B13h	4	DCLK_PMON_CTR2_LOW_REG on page 209	0h
B14-B17h	4	DCLK_PMON_CTR2_HIGH_REG on page 209	0h
B18-B1Bh	4	DCLK_PMON_CTR3_LOW_REG on page 209	0h
B1C-B1Fh	4	DCLK_PMON_CTR3_HIGH_REG on page 209	0h
B20-B23h	4	DCLK_PMON_CTRCTLO_REG on page 209	0h
<i>continued...</i>			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B24–B27h	4	DCLK_PMON_CTRCTL1_REG on page 210	0h
B28–B2Bh	4	DCLK_PMON_CTRCTL2_REG on page 211	0h
B2C–B2Fh	4	DCLK_PMON_CTRCTL3_REG on page 212	0h
B30–B33h	4	DCLK_PMON_UNIT_CTL_REG on page 213	0h
B34–B37h	4	DCLK_PMON_UNIT_STATUS_REG on page 214	0h
B3C–B3Fh	4	DCLK_PMON_TIMESTAMP_LOW_REG on page 214	0h
B40–B43h	4	DCLK_PMON_TIMESTAMP_HIGH_REG on page 214	0h
B44–B47h	4	DCLK_PMON_TIMESTAMP_CTL_REG on page 215	1h

3.7.1 DCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 4		Offset: B00	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr0_low — Dclk PMON Counter0 low 32 bits.				

3.7.2 DCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 4		Offset: B04	
Bit	Attr	Default	Description				
31:16	RO	0h	Reserved (RSVD) — Reserved.				
15:0	RW_V	0h	cr_dclk_pmon_ctr0_high — Dclk PMON Counter0 high 16 bits.				

3.7.3 DCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 9		Function: 4		Offset: B08	
Bit	Attr	Default	Description				
31:0	RW_V	0h	cr_dclk_pmon_ctr1_low — Dclk PMON Counter1 low 32 bits.				

3.7.4 DCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter1. It can also be used to load a value for testing.



Bus: 2		Device: 9	Function: 4	Offset: B0C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr1_high — Dclk PMON Counter1 high 16 bits.	

3.7.5 DCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 4	Offset: B10
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr2_low — Dclk PMON Counter2 low 32 bits.	

3.7.6 DCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 4	Offset: B14
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr2_high — Dclk PMON Counter2 high 16 bits.	

3.7.7 DCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 4	Offset: B18
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_ctr3_low — Dclk PMON Counter3 low 32 bits.	

3.7.8 DCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC DCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 9	Function: 4	Offset: B1C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_ctr3_high — Dclk PMON Counter3 high 16 bits.	

3.7.9 DCLK_PMON_CTRCTL0_REG

This register controls the operation of the Dclk PMON Counter0.



Bus: 2		Device: 9	Function: 4	Offset: B20
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl0_thresh — Dclk PMON Counter0 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl0_inv — Dclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl0_en — Dclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl0_oven — Dclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl0_ed — Dclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl0_rst — Dclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl0_umask — Dclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl0_evsel — Dclk PMON Counter0 Event Select. Selects the event to be counted.	

3.7.10 DCLK_PMON_CTLCTL1_REG

This register controls the operation of the Dclk PMON Counter1.



Bus: 2		Device: 9	Function: 4	Offset: B24
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl1_thresh — Dclk PMON Counter1 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl1_inv — Dclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl1_en — Dclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl1_oven — Dclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl1_ed — Dclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl1_rst — Dclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl1_umask — Dclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl1_evsel — Dclk PMON Counter1 Event Select. Selects the event to be counted.	

3.7.11 DCLK_PMON_CTL2_REG

This register controls the operation of the Dclk PMON Counter2.



Bus: 2		Device: 9	Function: 4	Offset: B28
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl2_thresh — Dclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl2_inv — Dclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl2_en — Dclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl2_oven — Dclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl2_ed — Dclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl2_rst — Dclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl2_umask — Dclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl2_evsel — Dclk PMON Counter2 Event Select. Selects the event to be counted.	

3.7.12 DCLK_PMON_CTL3_REG

This register controls the operation of the Dclk PMON Counter3.



Bus: 2		Device: 9	Function: 4	Offset: B2C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_dclk_pmon_ctl3_thresh — Dclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_dclk_pmon_ctl3_inv — Dclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_dclk_pmon_ctl3_en — Dclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_dclk_pmon_ctl3_oven — Dclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_dclk_pmon_ctl3_ed — Dclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_dclk_pmon_ctl3_rst — Dclk PMON Counter3 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_dclk_pmon_ctl3_umask — Dclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_dclk_pmon_ctl3_evsel — Dclk PMON Counter3 Event Select. Selects the event to be counted.	

3.7.13 DCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 4	Offset: B30
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_dclk_pmon_boxctl_freeze — Dclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
continued...				



Bus: 2		Device: 9	Function: 4	Offset: B30
Bit	Attr	Default	Description	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_dclk_pmon_boxctl_rstctr — Dclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_dclk_pmon_boxctl_rstcfg — Dclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.7.14 DCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Dclk PMON.

Bus: 2		Device: 9	Function: 4	Offset: B34
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Dclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Dclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Dclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Dclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	

3.7.15 DCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.

Bus: 2		Device: 9	Function: 4	Offset: B3C
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_dclk_pmon_timestamp_low — Dclk PMON Timestamp Counter low 32 bits.	

3.7.16 DCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of dclk cycles since reset was deasserted.



Bus: 2		Device: 9	Function: 4	Offset: B40
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_dclk_pmon_timestamp_high — Dclk PMON Timestamp Counter high 16 bits.	

3.7.17 DCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Dclk timestamp counter.

Bus: 2		Device: 9	Function: 4	Offset: B44
Bit	Attr	Default	Description	
31:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.	

3.8 Bus: 2, Device: 11, Function: 0 (CFG)

Table 32. Summary of Bus: 2, Device: 11, Function: 0 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
400–403h	4	UCLK_PMON_CTR0_LOW_REG on page 216	0h
404–407h	4	UCLK_PMON_CTR0_HIGH_REG on page 216	0h
408–40Bh	4	UCLK_PMON_CTR1_LOW_REG on page 216	0h
40C–40Fh	4	UCLK_PMON_CTR1_HIGH_REG on page 216	0h
410–413h	4	UCLK_PMON_CTR2_LOW_REG on page 216	0h
414–417h	4	UCLK_PMON_CTR2_HIGH_REG on page 217	0h
418–41Bh	4	UCLK_PMON_CTR3_LOW_REG on page 217	0h
41C–41Fh	4	UCLK_PMON_CTR3_HIGH_REG on page 217	0h
420–423h	4	UCLK_PMON_CTRCTL0_REG on page 217	0h
424–427h	4	UCLK_PMON_CTRCTL1_REG on page 218	0h
428–42Bh	4	UCLK_PMON_CTRCTL2_REG on page 219	0h
42C–42Fh	4	UCLK_PMON_CTRCTL3_REG on page 220	0h
430–433h	4	UCLK_PMON_UNIT_CTL_REG on page 221	0h
434–437h	4	UCLK_PMON_UNIT_STATUS_REG on page 221	0h
44C–44Fh	4	UCLK_PMON_TIMESTAMP_LOW_REG on page 222	0h
450–453h	4	UCLK_PMON_TIMESTAMP_HIGH_REG on page 222	0h
454–457h	4	UCLK_PMON_TIMESTAMP_CTL_REG on page 222	1h



3.8.1 UCLK_PMON_CTR0_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 11		Function: 0	Offset: 400
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr0_low — Uclk PMON Counter0 low 32 bits.		

3.8.2 UCLK_PMON_CTR0_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter0. It can also be used to load a value for testing.

Bus: 2		Device: 11		Function: 0	Offset: 404
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr0_high — Uclk PMON Counter0 high 16 bits.		

3.8.3 UCLK_PMON_CTR1_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 11		Function: 0	Offset: 408
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr1_low — Uclk PMON Counter1 low 32 bits.		

3.8.4 UCLK_PMON_CTR1_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter1. It can also be used to load a value for testing.

Bus: 2		Device: 11		Function: 0	Offset: 40C
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_ctr1_high — Uclk PMON Counter1 high 16 bits.		

3.8.5 UCLK_PMON_CTR2_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 11		Function: 0	Offset: 410
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_ctr2_low — Uclk PMON Counter2 low 32 bits.		



3.8.6 UCLK_PMON_CTR2_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter2. It can also be used to load a value for testing.

Bus: 2		Device: 11	Function: 0	Offset: 414
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr2_high — Uclk PMON Counter2 high 16 bits.	

3.8.7 UCLK_PMON_CTR3_LOW_REG

This register reflects the value of the lower 32 bits of the 48-bit MC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 11	Function: 0	Offset: 418
Bit	Attr	Default	Description	
31:0	RW_V	0h	cr_uclk_pmon_ctr3_low — Uclk PMON Counter3 low 32 bits.	

3.8.8 UCLK_PMON_CTR3_HIGH_REG

This register reflects the value of the upper 16 bits of the 48-bit MC UCLK PMON counter3. It can also be used to load a value for testing.

Bus: 2		Device: 11	Function: 0	Offset: 41C
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	cr_uclk_pmon_ctr3_high — Uclk PMON Counter3 high 16 bits.	

3.8.9 UCLK_PMON_CTRCTL0_REG

This register controls the operation of the Uclk PMON Counter0.

Bus: 2		Device: 11	Function: 0	Offset: 420
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl0_thresh — Uclk PMON Counter0 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl0_inv — Uclk PMON Counter0 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl0_en — Uclk PMON Counter0 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
continued...				



Bus: 2		Device: 11	Function: 0	Offset: 420
Bit	Attr	Default	Description	
20	RW_V	0h	cr_uclk_pmon_ctl0_oven — Uclk PMON Counter0 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl0_ed — Uclk PMON Counter0 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl0_rst — Uclk PMON Counter0 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl0_umask — Uclk PMON Counter0 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl0_evsel — Uclk PMON Counter0 Event Select. Selects the event to be counted.	

3.8.10 UCLK_PMON_CTL1_REG

This register controls the operation of the Uclk PMON Counter1.

Bus: 2		Device: 11	Function: 0	Offset: 424
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl1_thresh — Uclk PMON Counter1 Threshold. Sets the threshold on the the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl1_inv — Uclk PMON Counter1 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl1_en — Uclk PMON Counter1 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl1_oven — Uclk PMON Counter1 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The	
continued...				



Bus: 2		Device: 11	Function: 0	Offset: 424
Bit	Attr	Default	Description	
			time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl1_ed — Uclk PMON Counter1 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl1_rst — Uclk PMON Counter1 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl1_umask — Uclk PMON Counter1 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl1_evsel — Uclk PMON Counter1 Event Select. Selects the event to be counted.	

3.8.11 UCLK_PMON_CTL2_REG

This register controls the operation of the Uclk PMON Counter2.

Bus: 2		Device: 11	Function: 0	Offset: 428
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl2_thresh — Uclk PMON Counter2 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl2_inv — Uclk PMON Counter2 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl2_en — Uclk PMON Counter2 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl2_oven — Uclk PMON Counter2 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	

continued...



Bus: 2		Device: 11	Function: 0	Offset: 428
Bit	Attr	Default	Description	
18	RW_V	0h	cr_uclk_pmon_ctl2_ed — Uclk PMON Counter2 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl2_rst — Uclk PMON Counter2 Reset. Clears the counter to 0 when set.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl2_umask — Uclk PMON Counter2 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl2_evsel — Uclk PMON Counter2 Event Select. Selects the event to be counted.	

3.8.12 UCLK_PMON_CTL3_REG

This register controls the operation of the Uclk PMON Counter3.

Bus: 2		Device: 11	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
31:24	RW_V	0h	cr_uclk_pmon_ctl3_thresh — Uclk PMON Counter3 Threshold. Sets the threshold on the number of incoming events that need to be seen before the counter increments.	
23	RW_V	0h	cr_uclk_pmon_ctl3_inv — Uclk PMON Counter3 Invert. Inverts the selected event signal, so the counter counts when the event is not true. If edge detect is enabled, the counter counts falling edges if the Invert bit is set.	
22	RW_V	0h	cr_uclk_pmon_ctl3_en — Uclk PMON Counter3 Enable. Enables the counter to count events.	
21	RO	0h	Reserved (RSVD) — Reserved.	
20	RW_V	0h	cr_uclk_pmon_ctl3_oven — Uclk PMON Counter3 Overflow Enable. When overflow enable bit is set and the counter hits all Fs, it sets the status bit and send the overflow info to UBOX (the central PMON control logic is located there). UBOX upon receiving this info through message channel, will send freeze signal wire to all PMON module across un-tile. The counters in all PMON will stop counting on receiving this signal. The time this signal reaches each PMON module is varying and before it gets the signal, the counter continues to wrap over and count. If overflow enable bit is not set, the counter continues to wrap around and count and not set the overflow bit.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	cr_uclk_pmon_ctl3_ed — Uclk PMON Counter3 Edge Detect Enable. When set, the counter counts rising edges of an event.	
17	RW_V	0h	cr_uclk_pmon_ctl3_rst — Uclk PMON Counter3 Reset. Clears the counter to 0 when set.	
continued...				



Bus: 2		Device: 11	Function: 0	Offset: 42C
Bit	Attr	Default	Description	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	cr_uclk_pmon_ctl3_umask — Uclk PMON Counter3 UMask. Used to selectively mask bits in the event select field.	
7:0	RW_V	0h	cr_uclk_pmon_ctl3_evsel — Uclk PMON Counter3 Event Select. Selects the event to be counted.	

3.8.13 UCLK_PMON_UNIT_CTL_REG

This register provides overall unit control of the MC Uclk PMON.

Bus: 2		Device: 11	Function: 0	Offset: 430
Bit	Attr	Default	Description	
31:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	cr_uclk_pmon_boxctl_freeze — Uclk PMON Freeze register. Setting this bit disables any further events from incrementing the counters.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	RW_V	0h	cr_uclk_pmon_boxctl_rstctrs — Uclk PMON Counter Reset register. Setting this bit clears all the counters to 0.	
0	RW_V	0h	cr_uclk_pmon_boxctl_rstcfg — Uclk PMON Config Reset register. Setting this bit clears all the counter control register settings to 0.	

3.8.14 UCLK_PMON_UNIT_STATUS_REG

This register provides unit status of the MC Uclk PMON.

Bus: 2		Device: 11	Function: 0	Offset: 434
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW_V	0h	cr_boxctl_status3 — Uclk PMON Config Status3 register. Set if PMON Counter3 overflows. Write 1 to clear.	
2	RW_V	0h	cr_boxctl_status2 — Uclk PMON Config Status2 register. Set if PMON Counter2 overflows. Write 1 to clear.	
1	RW_V	0h	cr_boxctl_status1 — Uclk PMON Config Status1 register. Set if PMON Counter1 overflows. Write 1 to clear.	
0	RW_V	0h	cr_boxctl_status0 — Uclk PMON Config Status0 register. Set if PMON Counter0 overflows. Write 1 to clear.	



3.8.15 UCLK_PMON_TIMESTAMP_LOW_REG

This register reflects the lower 32-bits of the 64-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 11		Function: 0	Offset: 44C
Bit	Attr	Default	Description		
31:0	RW_V	0h	cr_uclk_pmon_timestamp_low — Uclk PMON Timestamp Counter low 32 bits.		

3.8.16 UCLK_PMON_TIMESTAMP_HIGH_REG

This register reflects the upper 16-bits of the 48-bit timestamp counter that counts the number of uclk cycles since reset was deasserted.

Bus: 2		Device: 11		Function: 0	Offset: 450
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	cr_uclk_pmon_timestamp_high — Uclk PMON Timestamp Counter high 16 bits.		

3.8.17 UCLK_PMON_TIMESTAMP_CTL_REG

This register is used to control and test the Uclk timestamp counter.

Bus: 2		Device: 11		Function: 0	Offset: 454
Bit	Attr	Default	Description		
31:2	RO	0h	Reserved (RSVD) — Reserved.		
1	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.		
0	RW_V	1h	cr_timestamp_ctr_enable — Enables the timestamp counter. Enabled by default.		



4.0 M2PCIE

The M2PCI is the logic which interfaces the IIO modules to the mesh and includes the mesh stop

This chapter describes in detail about the M2PCIE Performance monitoring MSRs.

The following table [Table 33: M2PCIE Register ID/Name Mapping] contains a mapping of the register names used in this document corresponding to the legacy Perfmon register names used conventionally on other Intel® processors for convenience of the reader.

Table 33. M2PCIE Register ID/Name Mapping

Register ID	Legacy Perfmon ID
PMONCNTRLOWER0_0	M2PCIE_PCI_PMON_CTR0_LOW
PMONCNTRUPPER0_0	M2PCIE_PCI_PMON_CTR0_HIGH
PMONCNTRLOWER0_1	M2PCIE_PCI_PMON_CTR1_LOW
PMONCNTRUPPER0_1	M2PCIE_PCI_PMON_CTR1_HIGH
PMONCNTRLOWER0_2	M2PCIE_PCI_PMON_CTR2_LOW
PMONCNTRUPPER0_2	M2PCIE_PCI_PMON_CTR2_HIGH
PMONCNTRLOWER0_3	M2PCIE_PCI_PMON_CTR3_LOW
PMONCNTRUPPER0_3	M2PCIE_PCI_PMON_CTR3_HIGH
PMONCNTRCFG0_0	M2PCIE_PCI_PMON_CTL0
PMONCNTRCFG0_1	M2PCIE_PCI_PMON_CTL1
PMONCNTRCFG0_2	M2PCIE_PCI_PMON_CTL2
PMONCNTRCFG0_3	M2PCIE_PCI_PMON_CTL3
PMONUNITCTRL0	M2PCIE_PCI_PMON_BOX_CTL
PMONCTRSTATUS0	M2PCIE_PCI_PMON_BOX_STATUS

4.1 Bus: 2, Device: 12, Function: 1 (CFG)

Table 34. Summary of Bus: 2, Device: 12, Function: 1 (CFG)

Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
A0–A3h	4	PMONCNTRLOWER0_0 on page 224	0h
A4–A7h	4	PMONCNTRUPPER0_0 on page 224	0h
A8–ABh	4	PMONCNTRLOWER0_1 on page 224	0h
AC–AFh	4	PMONCNTRUPPER0_1 on page 224	0h
continued...			



Offset	Size (Bytes)	Register Name (Register Symbol)	Default Value
B0-B3h	4	PMONCNTRLOWER0_2 on page 225	0h
B4-B7h	4	PMONCNTRUPPER0_2 on page 225	0h
B8-BBh	4	PMONCNTRLOWER0_3 on page 225	0h
BC-BFh	4	PMONCNTRUPPER0_3 on page 225	0h
D8-DBh	4	PMONCNTRCFG0_0 on page 225	0h
DC-DFh	4	PMONCNTRCFG0_1 on page 226	0h
E0-E3h	4	PMONCNTRCFG0_2 on page 227	0h
E4-E7h	4	PMONCNTRCFG0_3 on page 228	0h
F4-F7h	4	PMONUNITCTRL0 on page 229	30000h
F8-FBh	4	PMONCTRSTATUS0 on page 230	0h

4.1.1 PMONCNTRLOWER0_0

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12		Function: 1	Offset: A0
Bit	Attr	Default	Description		
31:0	RW_V	0h	pmonctrdata — This is the current value of the counter.		

4.1.2 PMONCNTRUPPER0_0

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12		Function: 1	Offset: A4
Bit	Attr	Default	Description		
31:16	RO	0h	Reserved (RSVD) — Reserved.		
15:0	RW_V	0h	pmonctrdata — This is the current value of the counter.		

4.1.3 PMONCNTRLOWER0_1

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12		Function: 1	Offset: A8
Bit	Attr	Default	Description		
31:0	RW_V	0h	pmonctrdata — This is the current value of the counter.		

4.1.4 PMONCNTRUPPER0_1

This register is a perfmon counter. Software can both read it and write it.



Bus: 2		Device: 12	Function: 1	Offset: AC
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	pmonctrdata — This is the current value of the counter.	

4.1.5 PMONCNTRLOWER0_2

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12	Function: 1	Offset: B0
Bit	Attr	Default	Description	
31:0	RW_V	0h	pmonctrdata — This is the current value of the counter.	

4.1.6 PMONCNTRUPPER0_2

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12	Function: 1	Offset: B4
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	pmonctrdata — This is the current value of the counter.	

4.1.7 PMONCNTRLOWER0_3

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12	Function: 1	Offset: B8
Bit	Attr	Default	Description	
31:0	RW_V	0h	pmonctrdata — This is the current value of the counter.	

4.1.8 PMONCNTRUPPER0_3

This register is a perfmon counter. Software can both read it and write it.

Bus: 2		Device: 12	Function: 1	Offset: BC
Bit	Attr	Default	Description	
31:16	RO	0h	Reserved (RSVD) — Reserved.	
15:0	RW_V	0h	pmonctrdata — This is the current value of the counter.	

4.1.9 PMONCNTRCFG0_0

Perfmon Counter Control Register



Bus: 2		Device: 12	Function: 1	Offset: D8
Bit	Attr	Default	Description	
31:24	RW_V	0h	threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.	
23	RW_V	0h	invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.	
22	RW_V	0h	counteren — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.	
21	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
20	RW_V	0h	ovfenable — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	edgedet — Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L1 example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.	
17	WO	0h	counterreset — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	unitmask — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.	
7:0	RW_V	0h	evslct — This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective Perfmon documentation.	

4.1.10 PMONCNTRCFG0_1

Bus: 2		Device: 12	Function: 1	Offset: DC
Bit	Attr	Default	Description	
31:24	RW_V	0h	threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the	
continued...				



Bus: 2		Device: 12	Function: 1	Offset: DC
Bit	Attr	Default	Description	
			comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.	
23	RW_V	0h	invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.	
22	RW_V	0h	counteren — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.	
21	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
20	RW_V	0h	ovfenable — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	edgedet — Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L1 example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.	
17	WO	0h	counterreset — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	unitmask — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.	
7:0	RW_V	0h	evslct — This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective Perfmon documentation.	

4.1.11 PMONCNTRCFG0_2

Bus: 2		Device: 12	Function: 1	Offset: E0
Bit	Attr	Default	Description	
31:24	RW_V	0h	threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the	
continued...				



Bus: 2		Device: 12	Function: 1	Offset: E0
Bit	Attr	Default	Description	
			comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.	
23	RW_V	0h	invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.	
22	RW_V	0h	counteren — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.	
21	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
20	RW_V	0h	ovfenable — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	edgedet — Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L1 example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.	
17	WO	0h	counterreset — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	unitmask — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.	
7:0	RW_V	0h	evslct — This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective Perfmon documentation.	

4.1.12 PMONCNTRCFG0_3

Bus: 2		Device: 12	Function: 1	Offset: E4
Bit	Attr	Default	Description	
31:24	RW_V	0h	threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the	
continued...				



Bus: 2		Device: 12	Function: 1	Offset: E4
Bit	Attr	Default	Description	
			comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.	
23	RW_V	0h	invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.	
22	RW_V	0h	counteren — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.	
21	RSVD-P	0h	Reserved (RSVD-P) — Reserved - protected.	
20	RW_V	0h	ovfenable — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.	
19	RO	0h	Reserved (RSVD) — Reserved.	
18	RW_V	0h	edgedet — Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L1 example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.	
17	WO	0h	counterreset — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.	
16	RO	0h	Reserved (RSVD) — Reserved.	
15:8	RW_V	0h	unitmask — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.	
7:0	RW_V	0h	evsict — This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective Perfmon documentation.	

4.1.13 PMONUNITCTRL0

Unit Control

Bus: 2		Device: 12	Function: 1	Offset: F4
Bit	Attr	Default	Description	
31:18	RO	0h	Reserved (RSVD) — Reserved.	
17	RW	1h	overflowenable — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count.	
16	RW	1h	freezeenable — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal.	
continued...				



Bus: 2		Device: 12	Function: 1	Offset: F4
Bit	Attr	Default	Description	
15:9	RO	0h	Reserved (RSVD) — Reserved.	
8	RW_V	0h	freeze counters — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset.	
7:2	RO	0h	Reserved (RSVD) — Reserved.	
1	WO	0h	reset counters — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset.	
0	WO	0h	resetcounter configs — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters.	

4.1.14 PMONCTRSTATUS0

Pmon Counter Status

Bus: 2		Device: 12	Function: 1	Offset: F8
Bit	Attr	Default	Description	
31:4	RO	0h	Reserved (RSVD) — Reserved.	
3	RW1C	0h	counter3ovf — counter 3 overflowed	
2	RW1C	0h	counter2ovf — counter 2 overflowed	
1	RW1C	0h	counter1ovf — counter 1 overflowed	
0	RW1C	0h	counter0ovf — counter 0 overflowed	



5.0 Model Specific Registers (MSR)

This chapter describes the Model Specific Registers (MSRs) in the processor.

The second column in the MSR descriptions titled "Scope" represents the scope of the bit field within the MSR as below:

- **Package:** The bit field is shared among the processor cores and execution threads in the same package.
- **Core:** If multiple cores exist within a processor package, then the bit field must be programmed once per core. The bit field is unique to each processor execution core within a package. Stated otherwise, these are per-core fields. When set to a particular value, it applies only to the core on which it was executed. A core scoped register is shared by all threads belonging to that core.
- **Thread:** If multiple threads exist within a processor core, then the bit field must be programmed once per thread.
- **Tile:** Two cores sharing the same second level cache.

All MSRs are 64-bit and accessed as a Qword quantity through the RDMSR and WRMSR instructions.

Please note that if not mentioned explicitly, the bits 32-63 for these registers should be considered as "Reserved (RSVD)"

The MSRs in this chapter are categorized into 2 sub- sections, namely

- **Core MSRs:** Describes in detail about the tile performance monitoring registers.
- **Uncore MSRs:** Describes in detail the performance monitoring registers part of the Untile components including UBOX, CHA and PCU,
 - **UBOX**
The UBox handles transactions like register accesses, interrupt flows, lock flows and events. The UBox also serves as the coordination point for the PCU's power management flows. In addition, the UBox houses coordination for the performance monitor architecture, scratch pad registers and semaphore registers for use by uCode and software.
 - **Caching and home agent or CHA**
The processor uncore is an LLC-less design and integrates the Home Agent (HA) functionality into the Caching Agent (CA). The Caching and Home Agent manages the interface between the processor cores, the mesh interconnect and the snoop filter. To support Knights Landing processor's non-inclusive Cache hierarchy feature, each CHA includes a new Snoop filter structure which tracks ownership of the cache lines inside the cores.
 - **Power Control Unit or PCU**
The processor's Power Control Unit (PCU) is a dedicated controller that provides power and thermal management for the processor. The PCU implements a PECI interface for out-of-band management. The PCU consists



of a dedicated microcontroller, ROM and RAM for pcode (PCU microcode), HW state machines; I/O registers for interfacing to the microcontroller and interfaces to the hardware units in the processor.

The following table [Table 35: UBOX and CHA0 Register ID/Name Mapping] contains a mapping of the UBOX and CHA0 register names used in this document corresponding to the legacy Perfmon register names used conventionally on other Intel® processors for convenience of the reader. Please note that the same mapping applies to the rest of the CHA1-37 registers.

NOTE: For the most updated definition of the PERF_UNIT_CTL_1_CHA_X register, please refer to Table 1-2 in the Intel® Xeon Phi™ processor Performance Monitoring Reference Manual—Volume 2: Events document located [here](#). Also refer to the Table 1-1. Bit fields of the MSR_OFFCORE_RESP {0, 1} Registers of the Intel® Xeon Phi™ Processor Performance Monitoring Reference Manual—Volume 2 : Events document for the latest definition of the MSR_OFFCORE_RESP {0, 1} Registers.

Table 35. UBOX and CHA0 Register ID/Name Mapping

Register ID	Legacy Perfmon ID
NcuPMONGICtrl	U_MSR_PMON_GLOBAL_CTRL
NcuPMONGISts	U_MSR_PMON_GLOBAL_STATUS
NCUPMONConfig	U_MSR_PMON_GLOBAL_CONFIG
NcuPMONCtlFix	U_MSR_PMON_UCLK_FIXED_CTL
NcuPMONCtrFx	U_MSR_PMON_UCLK_FIXED_CTR
NcuPMONCtCtl1	U_MSR_PMON_CTRL0
NcuPMONCtCtl2	U_MSR_PMON_CTRL1
NcuPMONCtSts	U_MSR_PMON_BOX_STATUS
NcuPMONCt1	U_MSR_PMON_CTR0
NcuPMONCt2	U_MSR_PMON_CTR1
PERF_CTR_UNIT_CTRL_CHA_0	CHA0_MSR_PMON_BOX_CTL
PERF_EVT_SEL_0_CHA_0	CHA0_MSR_PMON_CTL0
PERF_EVT_SEL_1_CHA_0	CHA0_MSR_PMON_CTL1
PERF_EVT_SEL_2_CHA_0	CHA0_MSR_PMON_CTL2
PERF_EVT_SEL_3_CHA_0	CHA0_MSR_PMON_CTL3
PERF_UNIT_CTL_CHA_0	CHA0_MSR_PMON_BOX_FILTER0
PERF_UNIT_CTL_1_CHA_0	CHA0_MSR_PMON_BOX_FILTER1
PERF_UNIT_STATUS_CHA_0	CHA0_MSR_PMON_BOX_STATUS
PERF_CTR_0_CHA_0	CHA0_MSR_PMON_CTR0
PERF_CTR_1_CHA_0	CHA0_MSR_PMON_CTR1
PERF_CTR_2_CHA_0	CHA0_MSR_PMON_CTR2
PERF_CTR_3_CHA_0	CHA0_MSR_PMON_CTR3

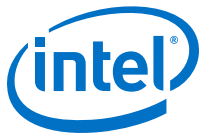
To get a detailed account of the Core and the Uncore MSRs related to performance monitoring, refer to the following sections.



5.1 Core MSRs

Table 36. Summary of Core MSR Registers

Offset Start	Register ID
C1h	(C1h) IA32_PMC0
C2h	(C2h) IA32_PMC1
186h	(186h) IA32_PERFEVTSEL0
187h	(187h) IA32_PERFEVTSEL1
1A0h	(1A0h) IA32_MISC_ENABLES
1A6h	(1A6h) MSR_OFFCORE_RESP0
1A7h	(1A7h) MSR_OFFCORE_RESP1
1C8h	(1C8h) MSR_LBR_SELECT
1C9h	(1C9h) MSR_LASTBRANCH_TOS
1D9h	(1D9h) IA32_DEBUGCTL
309h	(309h) IA32_FIXED_CTR0
30Ah	(30Ah) IA32_FIXED_CTR1
30Bh	(30Bh) IA32_FIXED_CTR2
345h	(345h) IA32_PERF_CAPABILITIES
38Dh	(38Dh) IA32_FIXED_CTR_CTRL
38Eh	(38Eh) IA32_PERF_GLOBAL_STATUS
38Fh	(38Fh) IA32_PERF_GLOBAL_CTRL
390h	(390h) IA32_PERF_GLOBAL_OVF_CTRL
3F1h	(3F1h) IA32_PEBS_ENABLE
483h	(483h) IA32_VMX_EXIT_CTL5
484h	(484h) IA32_VMX_ENTRY_CTL5
4C1h	(4C1h) IA32_A_PMC0
4C2h	(4C2h) IA32_A_PMC1
680h	(680h) MSR_LASTBRANCH_0_FROM_IP
681h	(681h) MSR_LASTBRANCH_1_FROM_IP
682h	(682h) MSR_LASTBRANCH_2_FROM_IP
683h	(683h) MSR_LASTBRANCH_3_FROM_IP
684h	(684h) MSR_LASTBRANCH_4_FROM_IP
685h	(685h) MSR_LASTBRANCH_5_FROM_IP
686h	(686h) MSR_LASTBRANCH_6_FROM_IP
687h	(687h) MSR_LASTBRANCH_7_FROM_IP
6C0h	(6C0h) MSR_LASTBRANCH_0_TO_IP
6C1h	(6C1h) MSR_LASTBRANCH_1_TO_IP
<i>continued...</i>	



Offset Start	Register ID
6C2h	(6C2h) MSR_LASTBRANCH_2_TO_IP
6C3h	(6C3h) MSR_LASTBRANCH_3_TO_IP
6C4h	(6C4h) MSR_LASTBRANCH_4_TO_IP
6C5h	(6C5h) MSR_LASTBRANCH_5_TO_IP
6C6h	(6C6h) MSR_LASTBRANCH_6_TO_IP
6C7h	(6C7h) MSR_LASTBRANCH_7_TO_IP

5.1.1 (C1h) IA32_PMC0

General purpose counter 0. It allows only 32 bit WRMSR.

MSR Address: C1h				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	GP_COUNTER0_VALUE —40-bit general counter 0. Only 32 bit write supported.

5.1.2 (C2h) IA32_PMC1

General purpose counter 1. It allows only 32 bit WRMSR.

MSR Address: C2h				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	GP_COUNTER1_VALUE —40-bit general counter 1. Only 32 bit writes supported.

5.1.3 (186h) IA32_PERFVTSEL0

Performance Event Selection Register 0.

MSR Address: 186h				
Bit	Scope	Default	Attribute	Description
63:32	-	-	-	RSVD_63_32 —Reserved
31:24	Thread	0h	RW	CMASK —When CMASK is not zero, the corresponding performance counter increments each cycle if the event count is greater than or equal to the CMASK.
23	Thread	0h	RW	INVI invert the CMASK.
22	Thread	0h	RW	EN —Enables the corresponding performance counter to commence counting when this bit is set.
21	Thread	0h	RW	ANY_THREAD —When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.
20	Thread	0h	RW	INT —Enables interrupt on counter overflow.
continued...				



MSR Address: 186h				
Bit	Scope	Default	Attribute	Description
19	Thread	0h	RW	PC —Enables pin control.
18	Thread	0h	RW	EDGE —Enables edge detection if set.
17	Thread	0h	RW	OS —Counts while in privilege level is ring 0.
16	Thread	0h	RW	USR —Counts while in privilege level is not ring 0.
15:8	Thread	0h	RW	UMASK —Qualifies the micro architectural condition to detect on the selected event logic.
7:0	Thread	0h	RW	EVENT_SELECT —Selects a performance event logic unit.

5.1.4 (187h) IA32_PERFEVTSEL1

Performance Event Selection Register 1

MSR Address: 187h				
Bit	Scope	Default	Attribute	Description
63:32	-	-	-	RSVD_63_32 —Reserved
31:24	Thread	0h	RW	CMASK —When CMASK is not zero, the corresponding performance counter increments each cycle if the event count is greater than or equal to the CMASK
23	Thread	0h	RW	INV —Invert the CMASK
22	Thread	0h	RW	EN —Enables the corresponding performance counter to commence counting when this bit is set
21	Thread	0h	RW	ANY_THREAD —When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR
20	Thread	0h	RW	INT —Enables interrupt on counter overflow
19	Thread	0h	RW	PC —Enables pin control
18	Thread	0h	RW	EDGE —Enables edge detection if set
17	Thread	0h	RW	OS —Counts while in privilege level is ring 0
16	Thread	0h	RW	USR —Counts while in privilege level is not ring 0
15:8	Thread	0h	RW	UMASK —Qualifies the microarchitectural condition to detect on the selected event logic
7:0	Thread	0h	RW	EVENT_SELECT — Selects a performance event logic unit

5.1.5 (1A0h) IA32_MISC_ENABLES

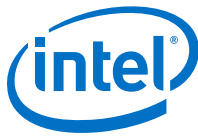
Miscellaneous Enables Register holds enable bits for miscellaneous features.



MSR Address: 1A0h				
Bit	Scope	Default	Attribute	Description
63:39	-	-	-	RSVD_63_39 —Reserved
38	Package	0h	RW	<p>TURBO_MODE_DISABLE—Disables turbo mode.</p> <p>If (CPUID.(EAX=6):EAX[1] == 0 and Turbo Mode Disable == 0)</p> <p>{</p> <p> Turbo Mode has been factory configured as disabled and is not available in this physical processor package. BIOS should not attempt to enable Turbo Mode via this MSR. BIOS should show Turbo Mode as Disabled and Not Configurable.</p> <p>}</p> <p>Else If (CPUID.(EAX=6):EAX[1] == 0 and Turbo Mode Disable == 1)</p> <p>{</p> <p> Turbo Mode is factory configured as available but globally disabled for the all logical processors in this processor package. BIOS can enable Turbo Mode by setting Turbo Mode Disable = 0.</p> <p>}</p> <p>Else If (CPUID.(EAX=6):EAX[1] == 1 and Turbo Mode Disable == 0)</p> <p>{</p> <p> Turbo Mode is factory-configured as available and enabled for all logical processors in this processor package.</p> <p>}</p> <p>More details on detecting and configuring Turbo Mode are available in the Turbo Mode section of the Processor BIOS Writer's Guide.</p> <p>On Reset, Turbo Mode Disable is set to 0 if the Turbo Mode feature is unavailable, and is set to 1 if Turbo Mode feature is available. This is done in order to guarantee that the processor does not go into Turbo Mode until BIOS has had an opportunity to enable this feature.</p> <p>After reset, Turbo Mode Disable is Read-Only if the Turbo Mode feature is unavailable, and read/write is the Turbo Mode feature is available.</p> <p>Turbo Mode Disable is also used to hide the Turbo Mode feature in the CPUID.(EAX=6) feature flag.</p> <p>If the Turbo Mode feature is available, CPUID.(EAX=6):EAX[2] Turbo Mode feature flag is always the complement of this package-level Turbo Mode Disable.</p>
37:35	-	-	-	RSVD_37_35 —Reserved
34	Thread	0h	RW	<p>XD_DIS—When set to a 1, disables the Execute Disable Bit feature (XD Bit). When set to a 1 the XD Bit extended feature flag (EDX[20]) is cleared to a 0 when the CPUID.(EAX = 80000001h).</p> <p>When set to a 0 (default) the Execute Disable Bit feature (if available) allows the OS to enable PAE paging and take advantage of data only pages.</p> <p>Note: Assuming this bit is not already set to a 1, if the XD Bit extended feature flag, EDX[20], is set to a 0 after executing CPUID.(80000001h), then this feature is not supported and BIOS must not alter the contents of this bit location. Writing this bit to a 1 when the XD Bit extended feature flag is set to 0 may generate a #GP exception.</p> <p>BIOS must restore this bit when the system comes out of S3 or S4.</p>
33:24	-	-	-	RSVD_33_24 —Reserved
continued...				



MSR Address: 1A0h				
Bit	Scope	Default	Attribute	Description
23	Tile	1h	RW	<p>TPR_MESSAGE_DISABLE—xTPR messages are optional messages that allow the processor to inform the chipset of its priority. When set to a 0, xTPR messages are transmitted on the system bus to the central agent. When set to a 1 (default), xTPR messages are not transmitted on the system bus.</p> <p>Note: The BIOS must leave this bit in its power-on default state for all platforms.</p>
22	Thread	0h	RW	<p>LIMIT_CPUID_MAXVAL—When set to 1, this causes the CPUID.(EAX=0):EAX[7:0] to return a maximum value of 3. When set to a 0 (default) the CPUID.(EAX=0) returns the number corresponding to the maximum standard function supported. Some OSs like Windows* NT cannot handle a Maxval > 3.</p> <p>BIOS must contain a setup option that allows the user to specify when an older OS is installed that does not support CPUID function > 3.</p> <p>Note: Before setting this bit, BIOS must execute the CPUID.(EAX=0):EAX[7:0] to get the maximum value. If the maximum value is > 3, then this bit is supported, otherwise this bit is not supported, and BIOS must not alter the contents of this bit location. Writing this bit when the maximum value is < 3 may generate a #GP exception.</p>
21:19	-	-	-	RSVD_21_19 —Reserved
18	Thread	1h	RW	<p>ENABLE_MONITOR_FSM—When set to 0, the MONITOR feature flag returned by the CPUID.(EAX=1):ECX[3] will be cleared indicating that the MONITOR and MWAIT instructions are not supported. An Invalid Opcode (#UD#) Exception will be generated if software attempts to execute either the MONITOR or MWAIT instruction when this bit is 0.</p> <p>It is recommended that BIOS not modify this bit.</p>
17	-	-	-	RSVD_17_17 —Reserved
16	Package	0h	RW	EIST_ENABLE —Setting this bit enables the Enhanced Intel SpeedStep Technology mechanism.
15:13	-	-	-	RSVD_15_13 —Reserved
12	Core	0h	RO	PEBS_UNAVAILABLE —When set, the processor does not support precise event-based sampling (PEBS); when clear, PEBS is supported.
11	Thread	0h	RO	BTS_UNAVAILABLE —When set, the processor does not support branch trace storage (BTS); when clear, BTS is supported.
10:8	-	-	-	RSVD_10_8 —Reserved
7	Core	1h	RO	PERFMON_AVAILABLE —When set, performance monitoring is enabled; when clear, performance monitoring is disabled.
6:4	-	-	-	RSVD_6_4 —Reserved
continued...				



MSR Address: 1A0h				
Bit	Scope	Default	Attribute	Description
3	Package	1h	RW	ADAPTIVE_TM_ENABLE —Enabling this bit allows Intel Adaptive Thermal monitor to function. This includes TM1, TM2 and EMTTM. System BIOS must set this bit to be operating within spec.
2:1	-	-	-	RSVD_2_1 —Reserved
0	Thread	1h	RW	FAST_STRINGS_EN —Setting this bit enables fast strings for REP MOVS and REP STOS. A value = 0 indicates disabled, and a value = 1 (default) indicates enabled. Note: System BIOS must leave this bit in default enabled state.

5.1.6 (1A6h) MSR_OFFCORE_RESP0

Used to program general purpose counter x when IA32_PERFVTSELx.
{EVENT_SELECT, UMASK} = {0xB7, 0x01}

MSR Address: 1A6h				
Bit	Scope	Default	Attribute	Description
63:39	-	-	-	RSVD_63_39 —Reserved
38	Tile	0h	RW	AVG_LATENCY
37	Tile	0h	RW	NON_DRAM_ADDR - Non-DRAM Address
36	Tile	0h	RW	SNOOP_HITM - Snoop Hit Modified
35	Tile	0h	RW	SNOOP_HIT_W_FWD - Snoop Hit, and Forwarding of data
34	Tile	0h	RW	SNOOP_HIT_NO_FWD - Snoop Hit, but no Forwarding of data
33	Tile	0h	RW	SNOOP_MISS - Snoop missed
32	Tile	0h	RW	SNOOP_NOT_NEEDED - Snoops were not needed
31	Tile	0h	RW	SNOOP_NONE - No cores were snooped
30:29	-	-	-	RSVD_30_29 —Reserved
28	Tile	0h	RW	L2_HIT_F_THIS
27	Tile	0h	RW	L2_HIT_S_THIS
26	Tile	0h	RW	L2_HIT_E_THIS
25	Tile	0h	RW	L2_HIT_M_THIS
24	Tile	0h	RW	HIT_DDR4_FAR
23	Tile	0h	RW	HIT_DDR4_LOCAL
22	Tile	0h	RW	HIT_MCDRAM_FAR - Hit MCDRAM in Far Cluster or Hit Far Cluster's L2/Snoop Filter
21	Tile	0h	RW	HIT_MCDRAM_LOCAL
20	Tile	0h	RW	L2_HITF_OTHER_TILE_NEAR - Other Tile's L2 or Snoop Filter Hit F in Near Cluster
19	Tile	0h	RW	L2_HITE_OTHER_TILE_NEAR - Other Tile's L2 or Snoop Filter Hit E in Near Cluster
continued...				

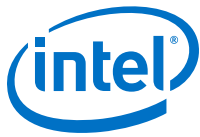


MSR Address: 1A6h				
Bit	Scope	Default	Attribute	Description
18	-	-	-	RSVD_18_18—Reserved
17	Tile	0h	RW	SUPPLIER_NONE
16	Tile	0h	RW	ANY_RESP
15	Tile	0h	RW	ANY_REQ
14	Tile	0h	RW	PARTIAL_STREAMING_STORE
13	Tile	0h	RW	DCU_HW_PREFETCH_DATA_RD
12	Tile	0h	RW	SW_PREFETCH
11	Tile	0h	RW	FULL_STREAMING_STORE
10	Tile	0h	RW	BUS_LOCK
9	Tile	0h	RW	UC_CODE_READ
8	Tile	0h	RW	PARTIAL_WRITE
7	Tile	0h	RW	PARTIAL_READ
6	Tile	0h	RW	L2_PREFETCH_CODE
5	Tile	0h	RW	L2_PREFETCH_RFO
4	Tile	0h	RW	L2_PREFETCH_LOAD
3	Tile	0h	RW	WRITEBACK
2	Tile	0h	RW	DEMAND_CODE
1	Tile	0h	RW	DEMAND_RFO
0	Tile	0h	RW	DEMAND_READ

5.1.7 (1A7h) MSR_OFFCORE_RESP1

Used to program general purpose counter x when IA32_PERFEVTSELx.
{EVENT_SELECT, UMASK} = {0xB7, 0x02}.

MSR Address: 1A7h				
Bit	Scope	Default	Attribute	Description
63:38	-	-	-	RSVD_63_38—Reserved
37	Tile	0h	RW	NON_DRAM_ADDR - Non-DRAM Address
36	Tile	0h	RW	SNOOP_HITM - Snoop Hit Modified
35	Tile	0h	RW	SNOOP_HIT_W_FWD - Snoop Hit, and Forwarding of data
34	Tile	0h	RW	SNOOP_HIT_NO_FWD - Snoop Hit, but no Forwarding of data
33	Tile	0h	RW	SNOOP_MISS - Snoop missed
32	Tile	0h	RW	SNOOP_NOT_NEEDED - Snoops were not needed
31	Tile	0h	RW	SNOOP_NONE - No cores were snooped
30:29	-	-	-	RSVD_30_29—Reserved
28	Tile	0h	RW	L2_HIT_F_THIS
continued...				



MSR Address: 1A7h				
Bit	Scope	Default	Attribute	Description
27	Tile	0h	RW	L2_HIT_S_THIS
26	Tile	0h	RW	L2_HIT_E_THIS
25	Tile	0h	RW	L2_HIT_M_THIS
24	Tile	0h	RW	HIT_DDR4_FAR
23	Tile	0h	RW	HIT_DDR4_LOCAL
22	Tile	0h	RW	HIT_MCDRAM_FAR - Hit MCDRAM in Far Cluster or Hit Far Cluster's L2/Snoop Filter
21	Tile	0h	RW	HIT_MCDRAM_LOCAL
20	Tile	0h	RW	L2_HITF_OTHER_TILE_NEAR - Other Tile's L2 or Snoop Filter Hit F in Near Cluster
19	Tile	0h	RW	L2_HITE_OTHER_TILE_NEAR - Other Tile's L2 or Snoop Filter Hit E in Near Cluster
18	-	-	-	RSVD_18_18 —Reserved
17	Tile	0h	RW	SUPPLIER_NONE
16	Tile	0h	RW	ANY_RESP
15	Tile	0h	RW	ANY_REQ
14	Tile	0h	RW	PARTIAL_STREAMING_STORE
13	Tile	0h	RW	DCU_HW_PREFETCH_DATA_RD
12	Tile	0h	RW	SW_PREFETCH
11	Tile	0h	RW	FULL_STREAMING_STORE
10	Tile	0h	RW	BUS_LOCK
9	Tile	0h	RW	UC_CODE_READ
8	Tile	0h	RW	PARTIAL_WRITE
7	Tile	0h	RW	PARTIAL_READ
6	Tile	0h	RW	L2_PREFETCH_CODE
5	Tile	0h	RW	L2_PREFETCH_RFO
4	Tile	0h	RW	L2_PREFETCH_LOAD
3	Tile	0h	RW	WRITEBACK
2	Tile	0h	RW	DEMAND_CODE
1	Tile	0h	RW	DEMAND_RFO
0	Tile	0h	RW	DEMAND_READ

5.1.8 (1C8h) MSR_LBR_SELECT

MSR_LBR_SELECT provides bit fields to specify the conditions of subsets of branches that will not be captured in the LBR



MSR Address: 1C8h				
Bit	Scope	Default	Attribute	Description
63:10	-	-	-	RSVD_63_10 —Reserved
9	-	-	-	RSVD_9_9 —Reserved
8	Thread	0h	RW	FAR_BRANCH —When set, do not capture far branches
7	Thread	0h	RW	NEAR_REL_JMP —When set, do not capture near relative jumps
6	Thread	0h	RW	NEAR_INDIRECT_JMP —When set, do not capture near indirect jumps
5	Thread	0h	RW	NEAR_RET —When set, do not capture near returns
4	Thread	0h	RW	NEAR_INDIRECT_CALL —When set, do not capture near indirect calls
3	Thread	0h	RW	NEAR_REL_CALL —When set, do not capture near relative calls
2	Thread	0h	RW	JCC —When set, do not capture conditional branches
1	Thread	0h	RW	CPL_NEQ_0 —When set, do not capture branches occurring in ring >0
0	Thread	0h	RW	CPL_EQ_0 —When set, do not capture branches occurring in ring 0

5.1.9 (1C9h) MSR_LASTBRANCH_TOS

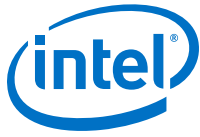
Contains a 3-bit pointer (0 ... 7) to the MSR in the LBR stack that contains the most recent branch, interrupt, or exception recorded. Prior to placing a new branch record on the stack, the TOS is incremented by 1. When the TOS pointer reaches 7, it wraps around to 0.

MSR Address: 1C9h				
Bit	Scope	Default	Attribute	Description
31:3	-	-	-	RSVD_31_3 —Reserved
2:0	Thread	0h	RW	TOS_POINTER —Pointer

5.1.10 (1D9h) IA32_DEBUGCTL

Trace/Profile Resource Control.

MSR Address: 1D9h				
Bit	Scope	Default	Attribute	Description
31:15	-	-	-	RSVD_31_15 —Reserved
14	Thread	0h	RW	FREEZE_WHILE_SMM_EN —When set, freezes perfmon and trace messages while in SMM
13	-	-	-	RESERVED_13_13 —Reserved
12	Thread	0h	RW	FREEZE_PERFMON_ON_PMI —When set, each ENABLE bit of the global counter control MSR are frozen (address 3BFH) on a PMI request
11	Thread	0h	RW	FREEZE_LBRS_ON_PMI —When set, the LBR stack is frozen on a PMI request
10	Thread	0h	RW	BTS_OFF_USR —When set, BTS or BTM is skipped if CPL > 0
9	Thread	0h	RW	BTS_OFF_OS —When set, BTS or BTM is skipped if CPL = 0
continued...				



MSR Address: 1D9h				
Bit	Scope	Default	Attribute	Description
8	Thread	0h	RW	BTINT —When clear, BTMs are logged in a BTS buffer in circular fashion. When this bit is set, an interrupt is generated by the BTS facility when the BTS buffer is full
7	Thread	0h	RW	BTS —Setting this bit enables branch trace messages (BTMs) to be logged in a BTS buffer
6	Thread	0h	RW	TR —Setting this bit to 1 enables branch trace messages to be sent
5:2	-	-	-	RSVD_5_2 —Reserved
1	Thread	0h	RW	BTF —Setting this bit to 1 enables the processor to treat EFLAGS.TF as single-step on branches instead of single-step on instructions
0	Thread	0h	RW	LBR —Setting this bit to 1 enables the processor to record a running trace of the most recent branches taken by the processor in the LBR stack

5.1.11 (309h) IA32_FIXED_CTR0

Fixed PerfMon counter 0 - Instructions retired.

MSR Address: 309h				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	FIXED_COUNTER0_VALUE —40-bit fixed counter 0

5.1.12 (30Ah) IA32_FIXED_CTR1

Fixed PerfMon counter 1 - Unhalted core cycles.

MSR Address: 30Ah				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	FIXED_COUNTER1_VALUE —40-bit fixed counter 1

5.1.13 (30Bh) IA32_FIXED_CTR2

Fixed PerfMon counter 2 - Unhalted reference cycles.

MSR Address: 30Bh				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	FIXED_COUNTER2_VALUE —40-bit fixed counter 2

5.1.14 (345h) IA32_PERF_CAPABILITIES

This register enumerates the existence of certain debug features in PerfMon.

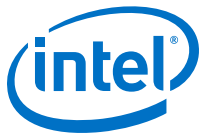


MSR Address: 345h				
Bit	Scope	Default	Attribute	Description
63:14	-	-	-	RSVD_63_14 —Reserved
13	Package	1h	RO	PMON_FULL_WIDTH_WRITE —Full width of counter writable via IA32_A_PMCx
12	Package	1h	RO	SMM_FREEZE —Freeze while SMM is supported
11:8	Package	2h	RO	PEBS_REC_FORMAT —PEBS Record Format
7	Package	1h	RO	PEBS_SAVE_ARCH_REGS —When set, PEBS will save architectural register and state information according to the encoded value of the PEBSRecordFormat field.
6	Package	1h	RO	PEBS_TRAP —PEBS Record Format. PEBS_TRAP = 1 denotes PEBS recording is trap-like. PEBS_TRAP = 0 denotes PEBS recording is fault-like.
5:0	Package	1h	RO	LBR_FORMAT —Last Branch Record Format. 0=32-bit, 1=64-bit LIP, 2=64-bit EIP

5.1.15 (38Dh) IA32_FIXED_CTR_CTRL

This register provides enables for the fixed performance counters and enables for PMI. This register also provides the control for the counts to be for single thread or both threads in a core.

MSR Address: 38Dh				
Bit	Scope	Default	Attribute	Description
31:12	-	-	-	RSVD_31_12 —Reserved
11	Thread	0h	RW	CTR_2_PMI_ENABLE —When set to 1 will cause a PMI when the 48 bit fixed counter 2 overflows.
10	Thread	0h	RW	CTR_2_ANY_THREAD_ENABLE —When this bit is set fixed counter 0 will count instructions retired for any thread.
9:8	Thread	0h	RW	CTR_2_ENABLE —This bit field is the local enable for the Fixed counter 2. Disabling these bits will cause the counters to halt. Counter is enabled by setting CPL Level: 00 - Disable 01 - Ring 0 (OS) 10 - Ring 1,2,3 (User) 11 - All ring levels (OS or User) when CPL changes to one of the above the counter gets enabled.
7	Thread	0h	RW	CTR_1_PMI_ENABLE —When set to 1 will cause a PMI when the 48 bit fixed counter 1 overflows.
6	Thread	0h	RW	CTR_1_ANY_THREAD_ENABLE —When this bit is set fixed counter 1 will count instructions retired for any thread.
5:4	Thread	0h	RW	CTR_1_ENABLE —This bit field is the local enable for the Fixed counter 1. Disabling these bits will cause the counters to halt. Counter is enabled by setting CPL Level: 00 - Disable 01 - Ring 0 (OS) 10 - Ring 1,2,3 (User) 11 - All ring levels (OS or User) when CPL changes to one of the above the counter gets enabled.
continued...				



MSR Address: 38Dh				
Bit	Scope	Default	Attribute	Description
3	Thread	0h	RW	CTR_0_PMI_ENABLE —When set to 1 will cause a PMI when the 48 bit fixed counter 0 overflows.
2	Thread	0h	RW	CTR_0_ANY_THREAD_ENABLE —When this bit is set fixed counter 0 will count instructions retired for any thread.
1:0	Thread	0h	RW	CTR0_ENABLE —This bit field is the local enable for the Fixed counter 0. Disabling these bits will cause the counters to halt. Counter is enabled by setting CPL Level: 00 - Disable 01 - Ring 0 (OS) 10 - Ring 1,2,3 (User) 11 - All ring levels (OS or User) when CPL changes to one of the above the counter gets enabled.

5.1.16 (38Eh) IA32_PERF_GLOBAL_STATUS

Global Performance Counter Status

MSR Address: 38Eh				
Bit	Scope	Default	Attribute	Description
63	Thread	0h	RO	COND_CHG — Status bits of this register has changed
62	Thread	0h	RO	OVF_BUF — DS SAVE area Buffer overflow status
61:35	-	-	-	RSVD_61_35 — Reserved
34	Thread	0h	RO	OVF_FIXED_CTR2 — Overflow status of IA32_FIXED_CTR2
33	Thread	0h	RO	OVF_FIXED_CTR1 — Overflow status of IA32_FIXED_CTR1
32	Thread	0h	RO	OVF_FIXED_CTR0 — Overflow status of IA32_FIXED_CTR0
31:2	-	-	-	RSVD_31_2 — Reserved
1	Thread	0h	RO	OVF_PMC1 — Overflow status of IA32_PMC1
0	Thread	0h	RO	OVF_PMC0 — Overflow status of IA32_PMC0

5.1.17 (38Fh) IA32_PERF_GLOBAL_CTRL

Global Perfmon Enable/Disable register

MSR Address: 38Fh				
Bit	Scope	Default	Attribute	Description
63:35	-	-	-	RSVD_63_35 —Reserved
34	Thread	0h	RW	ENABLE_FIXED_PERFMON_COUNTER_2
33	Thread	0h	RW	ENABLE_FIXED_PERFMON_COUNTER_1
32	Thread	0h	RW	ENABLE_FIXED_PERFMON_COUNTER_0
31:2	-	-	-	RSVD_31_2 —Reserved
1	Thread	1h	RW	ENABLE_GP_PERFMON_COUNTER_1
0	Thread	1h	RW	ENABLE_GP_PERFMON_COUNTER_0

5.1.18 (390h) IA32_PERF_GLOBAL_OVF_CTRL

Global Performance Overflow Counter Status



MSR Address: 390h				
Bit	Scope	Default	Attribute	Description
63	Thread	0h	WO	COND_CHG_CLR — Writes of '1' to this bit causes the 'Condition Change' bit in the IA32_PERF_GLOBAL_STATUS register to clear to '0'
62	Thread	0h	WO	OVF_BUF_CLEAR — Writes of '1' to this bit causes the 'Overflow pebs' bit in the global status register to clear to '0'
61:35	-	-	-	RSVD_61_35 —Reserved
34	Thread	0h	WO	OVF_FIXED_CTR2 — Set 1 to Clear OVF_FIXED_CTR2 bit
33	Thread	0h	WO	OVF_FIXED_CTR1 — Set 1 to Clear OVF_FIXED_CTR1 bit
32	Thread	0h	WO	OVF_FIXED_CTR0 — Set 1 to Clear OVF_FIXED_CTR0 bit
31:2	-	-	-	RSVD_31_2 —Reserved
1	Thread	0h	WO	OVF_PMC1 — Set 1 to Clear OVF_PMC1 bit
0	Thread	0h	WO	OVF_PMC0 — Set 1 to Clear OVF_PMC0 bit

5.1.19 (3F1h) IA32_PEBS_ENABLE

Precise Event Based Sampling controlling MSR

MSR Address: 3F1h				
Bit	Scope	Default	Attribute	Description
31:1	-	-	-	RSVD_31_1 —Reserved
0	Thread	0h	RW	PEBS_ENABLE —When set, enables Precise Event Based Sampling

5.1.20 (483h) IA32_VMX_EXIT_CTLs

Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual for a complete details on the VM-Exit Controls.

MSR Address: 483h				
Bit	Scope	Default	Attribute	Description
63:0	Core	-	-	DATA —Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual for a complete details on the VM-Exit Controls.

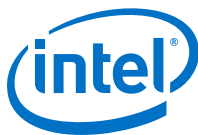
5.1.21 (484h) IA32_VMX_ENTRY_CTLs

Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual for a complete details on MSR related to the VM-Entry Controls.

MSR Address: 484h				
Bit	Scope	Default	Attribute	Description
63:0	Core	-	-	DATA —Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual for a complete details on MSR related to the VM-Entry Controls.

5.1.22 (4C1h) IA32_A_PMC0

This is an alias for the general purpose counter 0. It allows the full 40 bit WRMSR



MSR Address: 4C1h				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	GP_COUNTER0_VALUE —40-bit general counter 0. Full 40 bit writes supported.

5.1.23 (4C2h) IA32_A_PMC1

This is an alias for the general purpose counter 1. It allows the full 40 bit WRMSR

MSR Address: 4C2h				
Bit	Scope	Default	Attribute	Description
63:40	-	-	-	RSVD_63_40 —Reserved
39:0	Thread	0h	RW	GP_COUNTER1_VALUE —40-bit general counter 1. Full 40 bit writes supported.

5.1.24 (680h) MSR_LASTBRANCH_0_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 680h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.

5.1.25 (681h) MSR_LASTBRANCH_1_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 681h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.



5.1.26 (682h) MSR_LASTBRANCH_2_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 682h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.

5.1.27 (683h) MSR_LASTBRANCH_3_FROM_IP

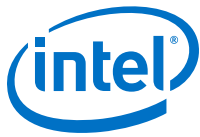
One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 683h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.

5.1.28 (684h) MSR_LASTBRANCH_4_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 684h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.



5.1.29 (685h) MSR_LASTBRANCH_5_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 685h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.

5.1.30 (686h) MSR_LASTBRANCH_6_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 686h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.

5.1.31 (687h) MSR_LASTBRANCH_7_FROM_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the source instruction for one of the last eight branches, exceptions, or interrupts taken by the processor. See also MSR_LASTBRANCH_TOS (1C9h)

MSR Address: 687h				
Bit	Scope	Default	Attribute	Description
63	Thread	-	RO	MISPREDICT —Mispredict bit - when set, indicates either the target of the branch was mispredicted and/or the direction (taken/nontaken) was mispredicted; otherwise, the target branch was predicted.
62:61	-	-	-	RSVD_62_61 —Reserved
60:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the branch instruction itself, this is the "branch from" address.



5.1.32 (6C0h) MSR_LASTBRANCH_0_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C0h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.33 (6C1h) MSR_LASTBRANCH_1_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C1h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA — The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.34 (6C2h) MSR_LASTBRANCH_2_TO_IP

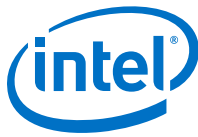
One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C2h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA — The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.35 (6C3h) MSR_LASTBRANCH_3_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C3h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA — The linear address of the target of the branch instruction itself, this is the "branch to" address.



5.1.36 (6C4h) MSR_LASTBRANCH_4_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C4h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA — The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.37 (6C5h) MSR_LASTBRANCH_5_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C5h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA — The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.38 (6C6h) MSR_LASTBRANCH_6_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C6h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the target of the branch instruction itself, this is the "branch to" address.

5.1.39 (6C7h) MSR_LASTBRANCH_7_TO_IP

One of eight pairs of last branch record registers on the last branch record stack. This part of the stack contains pointers to the destination instruction for one of the last eight branches, exceptions, or interrupts taken by the processor

MSR Address: 6C7h				
Bit	Scope	Default	Attribute	Description
63:48	Thread	-	RO	SIGN_EXT —Sign extension of bit 47 in this register
47:0	Thread	-	RO	DATA —The linear address of the target of the branch instruction itself, this is the "branch to" address.



5.2 Uncore MSRs

Table 37. Summary of Uncore MSR Registers

Offset Start	Register ID
700h	(700h) NcuPMONGICtrl on page 264
701h	(701h) NcuPMONGISts on page 269
702h	(702h) NCUPMONConfig on page 272
703h	(703h) NcuPMONCtIFix on page 272
704h	(704h) NcuPMONCtrFx on page 273
705h	(705h) NcuPMONCtCtl1 on page 273
706h	(706h) NcuPMONCtCtl2 on page 274
708h	(708h) NcuPMONCtSts on page 275
709h	(709h) NcuPMONCt1 on page 276
70Ah	(70Ah) NcuPMONCt2 on page 276
710h	(710h) PmonUnitCtrl on page 276
711h	(711h) PmonCntrCfg_0 on page 277
712h	(712h) PmonCntrCfg_1 on page 279
713h	(713h) PmonCntrCfg_2 on page 281
714h	(714h) PmonCntrCfg_3 on page 283
717h	(717h) PmonCntr_0 on page 285
718h	(718h) PmonCntr_1 on page 285
719h	(719h) PmonCntr_2 on page 285
71Ah	(71Ah) PmonCntr_3 on page 286
E00h	(E00h) PERF_CTR_UNIT_CTRL_CHA_0 on page 286
E01h	(E01h) PERF_EVT_SEL_0_CHA_0 on page 287
E02h	(E02h) PERF_EVT_SEL_1_CHA_0 on page 288
E03h	(E03h) PERF_EVT_SEL_2_CHA_0 on page 289
E04h	(E04h) PERF_EVT_SEL_3_CHA_0 on page 290
E05h	(E05h) PERF_UNIT_CTL_CHA_0 on page 292
E06h	(E06h) PERF_UNIT_CTL_1_CHA_0 on page 292
E07h	(E07h) PERF_UNIT_STATUS_CHA_0 on page 293
E08h	(E08h) PERF_CTR_0_CHA_0 on page 293
E09h	(E09h) PERF_CTR_1_CHA_0 on page 293
E0Ah	(E0Ah) PERF_CTR_2_CHA_0 on page 294
E0Bh	(E0Bh) PERF_CTR_3_CHA_0 on page 294
E0Ch	(E0Ch) PERF_CTR_UNIT_CTRL_CHA_1 on page 294
E0Dh	(E0Dh) PERF_EVT_SEL_0_CHA_1 on page 295
continued...	



Offset Start	Register ID
E0Eh	(E0Eh) PERF_EVT_SEL_1_CHA_1 on page 296
E0Fh	(E0Fh) PERF_EVT_SEL_2_CHA_1 on page 297
E10h	(E10h) PERF_EVT_SEL_3_CHA_1 on page 299
E11h	(E11h) PERF_UNIT_CTL_CHA_1 on page 300
E12h	(E12h) PERF_UNIT_CTL_1_CHA_1 on page 301
E13h	(E13h) PERF_UNIT_STATUS_CHA_1 on page 301
E14h	(E14h) PERF_CTR_0_CHA_1 on page 301
E15h	(E15h) PERF_CTR_1_CHA_1 on page 302
E16h	(E16h) PERF_CTR_2_CHA_1 on page 302
E17h	(E17h) PERF_CTR_3_CHA_1 on page 302
E18h	(E18h) PERF_CTR_UNIT_CTRL_CHA_2 on page 302
E19h	(E19h) PERF_EVT_SEL_0_CHA_2 on page 303
E1Ah	(E1Ah) PERF_EVT_SEL_1_CHA_2 on page 304
E1Bh	(E1Bh) PERF_EVT_SEL_2_CHA_2 on page 306
E1Ch	(E1Ch) PERF_EVT_SEL_3_CHA_2 on page 307
E1Dh	(E1Dh) PERF_UNIT_CTL_CHA_2 on page 308
E1Eh	(E1Eh) PERF_UNIT_CTL_1_CHA_2 on page 309
E1Fh	(E1Fh) PERF_UNIT_STATUS_CHA_2 on page 309
E20h	(E20h) PERF_CTR_0_CHA_2 on page 310
E21h	(E21h) PERF_CTR_1_CHA_2 on page 310
E22h	(E22h) PERF_CTR_2_CHA_2 on page 310
E23h	(E23h) PERF_CTR_3_CHA_2 on page 310
E24h	(E24h) PERF_CTR_UNIT_CTRL_CHA_3 on page 311
E25h	(E25h) PERF_EVT_SEL_0_CHA_3 on page 311
E26h	(E26h) PERF_EVT_SEL_1_CHA_3 on page 312
E27h	(E27h) PERF_EVT_SEL_2_CHA_3 on page 314
E28h	(E28h) PERF_EVT_SEL_3_CHA_3 on page 315
E29h	(E29h) PERF_UNIT_CTL_CHA_3 on page 316
E2Ah	(E2Ah) PERF_UNIT_CTL_1_CHA_3 on page 317
E2Bh	(E2Bh) PERF_UNIT_STATUS_CHA_3 on page 318
E2Ch	(E2Ch) PERF_CTR_0_CHA_3 on page 318
E2Dh	(E2Dh) PERF_CTR_1_CHA_3 on page 318
E2Eh	(E2Eh) PERF_CTR_2_CHA_3 on page 318
E2Fh	(E2Fh) PERF_CTR_3_CHA_3 on page 318
E30h	(E30h) PERF_CTR_UNIT_CTRL_CHA_4 on page 319
E31h	(E31h) PERF_EVT_SEL_0_CHA_4 on page 319
continued...	



Offset Start	Register ID
E32h	(E32h) PERF_EVT_SEL_1_CHA_4 on page 321
E33h	(E33h) PERF_EVT_SEL_2_CHA_4 on page 322
E34h	(E34h) PERF_EVT_SEL_3_CHA_4 on page 323
E35h	(E35h) PERF_UNIT_CTL_CHA_4 on page 325
E36h	(E36h) PERF_UNIT_CTL_1_CHA_4 on page 325
E37h	(E37h) PERF_UNIT_STATUS_CHA_4 on page 326
E38h	(E38h) PERF_CTR_0_CHA_4 on page 326
E39h	(E39h) PERF_CTR_1_CHA_4 on page 326
E3Ah	(E3Ah) PERF_CTR_2_CHA_4 on page 327
E3Bh	(E3Bh) PERF_CTR_3_CHA_4 on page 327
E3Ch	(E3Ch) PERF_CTR_UNIT_CTRL_CHA_5 on page 327
E3Dh	(E3Dh) PERF_EVT_SEL_0_CHA_5 on page 328
E3Eh	(E3Eh) PERF_EVT_SEL_1_CHA_5 on page 329
E3Fh	(E3Fh) PERF_EVT_SEL_2_CHA_5 on page 330
E40h	(E40h) PERF_EVT_SEL_3_CHA_5 on page 332
E41h	(E41h) PERF_UNIT_CTL_CHA_5 on page 333
E42h	(E42h) PERF_UNIT_CTL_1_CHA_5 on page 333
E43h	(E43h) PERF_UNIT_STATUS_CHA_5 on page 334
E44h	(E44h) PERF_CTR_0_CHA_5 on page 334
E45h	(E45h) PERF_CTR_1_CHA_5 on page 335
E46h	(E46h) PERF_CTR_2_CHA_5 on page 335
E47h	(E47h) PERF_CTR_3_CHA_5 on page 335
E48h	(E48h) PERF_CTR_UNIT_CTRL_CHA_6 on page 335
E49h	(E49h) PERF_EVT_SEL_0_CHA_6 on page 336
E4Ah	(E4Ah) PERF_EVT_SEL_1_CHA_6 on page 337
E4Bh	(E4Bh) PERF_EVT_SEL_2_CHA_6 on page 338
E4Ch	(E4Ch) PERF_EVT_SEL_3_CHA_6 on page 340
E4Dh	(E4Dh) PERF_UNIT_CTL_CHA_6 on page 341
E4Eh	(E4Eh) PERF_UNIT_CTL_1_CHA_6 on page 342
E4Fh	(E4Fh) PERF_UNIT_STATUS_CHA_6 on page 342
E50h	(E50h) PERF_CTR_0_CHA_6 on page 343
E51h	(E51h) PERF_CTR_1_CHA_6 on page 343
E52h	(E52h) PERF_CTR_2_CHA_6 on page 343
E53h	(E53h) PERF_CTR_3_CHA_6 on page 343
E54h	(E54h) PERF_CTR_UNIT_CTRL_CHA_7 on page 344
E55h	(E55h) PERF_EVT_SEL_0_CHA_7 on page 344
continued...	



Offset Start	Register ID
E56h	(E56h) PERF_EVT_SEL_1_CHA_7 on page 345
E57h	(E57h) PERF_EVT_SEL_2_CHA_7 on page 347
E58h	(E58h) PERF_EVT_SEL_3_CHA_7 on page 348
E59h	(E59h) PERF_UNIT_CTL_CHA_7 on page 349
E5Ah	(E5Ah) PERF_UNIT_CTL_1_CHA_7 on page 350
E5Bh	(E5Bh) PERF_UNIT_STATUS_CHA_7 on page 351
E5Ch	(E5Ch) PERF_CTR_0_CHA_7 on page 351
E5Dh	(E5Dh) PERF_CTR_1_CHA_7 on page 351
E5Eh	(E5Eh) PERF_CTR_2_CHA_7 on page 351
E5Fh	(E5Fh) PERF_CTR_3_CHA_7 on page 351
E60h	(E60h) PERF_CTR_UNIT_CTRL_CHA_8 on page 352
E61h	(E61h) PERF_EVT_SEL_0_CHA_8 on page 352
E62h	(E62h) PERF_EVT_SEL_1_CHA_8 on page 354
E63h	(E63h) PERF_EVT_SEL_2_CHA_8 on page 355
E64h	(E64h) PERF_EVT_SEL_3_CHA_8 on page 356
E65h	(E65h) PERF_UNIT_CTL_CHA_8 on page 358
E66h	(E66h) PERF_UNIT_CTL_1_CHA_8 on page 358
E67h	(E67h) PERF_UNIT_STATUS_CHA_8 on page 359
E68h	(E68h) PERF_CTR_0_CHA_8 on page 359
E69h	(E69h) PERF_CTR_1_CHA_8 on page 359
E6Ah	(E6Ah) PERF_CTR_2_CHA_8 on page 360
E6Bh	(E6Bh) PERF_CTR_3_CHA_8 on page 360
E6Ch	(E6Ch) PERF_CTR_UNIT_CTRL_CHA_9 on page 360
E6Dh	(E6Dh) PERF_EVT_SEL_0_CHA_9 on page 361
E6Eh	(E6Eh) PERF_EVT_SEL_1_CHA_9 on page 362
E6Fh	(E6Fh) PERF_EVT_SEL_2_CHA_9 on page 363
E70h	(E70h) PERF_EVT_SEL_3_CHA_9 on page 365
E71h	(E71h) PERF_UNIT_CTL_CHA_9 on page 366
E72h	(E72h) PERF_UNIT_CTL_1_CHA_9 on page 366
E73h	(E73h) PERF_UNIT_STATUS_CHA_9 on page 367
E74h	(E74h) PERF_CTR_0_CHA_9 on page 367
E75h	(E75h) PERF_CTR_1_CHA_9 on page 368
E76h	(E76h) PERF_CTR_2_CHA_9 on page 368
E77h	(E77h) PERF_CTR_3_CHA_9 on page 368
E78h	(E78h) PERF_CTR_UNIT_CTRL_CHA_10 on page 368
E79h	(E79h) PERF_EVT_SEL_0_CHA_10 on page 369
continued...	



Offset Start	Register ID
E7Ah	(E7Ah) PERF_EVT_SEL_1_CHA_10 on page 370
E7Bh	(E7Bh) PERF_EVT_SEL_2_CHA_10 on page 371
E7Ch	(E7Ch) PERF_EVT_SEL_3_CHA_10 on page 373
E7Dh	(E7Dh) PERF_UNIT_CTL_CHA_10 on page 374
E7Eh	(E7Eh) PERF_UNIT_CTL_1_CHA_10 on page 375
E7Fh	(E7Fh) PERF_UNIT_STATUS_CHA_10 on page 375
E80h	(E80h) PERF_CTR_0_CHA_10 on page 376
E81h	(E81h) PERF_CTR_1_CHA_10 on page 376
E82h	(E82h) PERF_CTR_2_CHA_10 on page 376
E83h	(E83h) PERF_CTR_3_CHA_10 on page 376
E84h	(E84h) PERF_CTR_UNIT_CTRL_CHA_11 on page 377
E85h	(E85h) PERF_EVT_SEL_0_CHA_11 on page 377
E86h	(E86h) PERF_EVT_SEL_1_CHA_11 on page 378
E87h	(E87h) PERF_EVT_SEL_2_CHA_11 on page 380
E88h	(E88h) PERF_EVT_SEL_3_CHA_11 on page 381
E89h	(E89h) PERF_UNIT_CTL_CHA_11 on page 382
E8Ah	(E8Ah) PERF_UNIT_CTL_1_CHA_11 on page 383
E8Bh	(E8Bh) PERF_UNIT_STATUS_CHA_11 on page 384
E8Ch	(E8Ch) PERF_CTR_0_CHA_11 on page 384
E8Dh	(E8Dh) PERF_CTR_1_CHA_11 on page 384
E8Eh	(E8Eh) PERF_CTR_2_CHA_11 on page 384
E8Fh	(E8Fh) PERF_CTR_3_CHA_11 on page 384
E90h	(E90h) PERF_CTR_UNIT_CTRL_CHA_12 on page 385
E91h	(E91h) PERF_EVT_SEL_0_CHA_12 on page 385
E92h	(E92h) PERF_EVT_SEL_1_CHA_12 on page 387
E93h	(E93h) PERF_EVT_SEL_2_CHA_12 on page 388
E94h	(E94h) PERF_EVT_SEL_3_CHA_12 on page 389
E95h	(E95h) PERF_UNIT_CTL_CHA_12 on page 391
E96h	(E96h) PERF_UNIT_CTL_1_CHA_12 on page 391
E97h	(E97h) PERF_UNIT_STATUS_CHA_12 on page 392
E98h	(E98h) PERF_CTR_0_CHA_12 on page 392
E99h	(E99h) PERF_CTR_1_CHA_12 on page 392
E9Ah	(E9Ah) PERF_CTR_2_CHA_12 on page 393
E9Bh	(E9Bh) PERF_CTR_3_CHA_12 on page 393
E9Ch	(E9Ch) PERF_CTR_UNIT_CTRL_CHA_13 on page 393
E9Dh	(E9Dh) PERF_EVT_SEL_0_CHA_13 on page 394
continued...	



Offset Start	Register ID
E9Eh	(E9Eh) PERF_EVT_SEL_1_CHA_13 on page 395
E9Fh	(E9Fh) PERF_EVT_SEL_2_CHA_13 on page 396
EA0h	(EA0h) PERF_EVT_SEL_3_CHA_13 on page 398
EA1h	(EA1h) PERF_UNIT_CTL_CHA_13 on page 399
EA2h	(EA2h) PERF_UNIT_CTL_1_CHA_13 on page 399
EA3h	(EA3h) PERF_UNIT_STATUS_CHA_13 on page 400
EA4h	(EA4h) PERF_CTR_0_CHA_13 on page 400
EA5h	(EA5h) PERF_CTR_1_CHA_13 on page 401
EA6h	(EA6h) PERF_CTR_2_CHA_13 on page 401
EA7h	(EA7h) PERF_CTR_3_CHA_13 on page 401
EA8h	(EA8h) PERF_CTR_UNIT_CTRL_CHA_14 on page 401
EA9h	(EA9h) PERF_EVT_SEL_0_CHA_14 on page 402
EAAh	(EAAh) PERF_EVT_SEL_1_CHA_14 on page 403
EABh	(EABh) PERF_EVT_SEL_2_CHA_14 on page 404
EACH	(EACH) PERF_EVT_SEL_3_CHA_14 on page 406
EADh	(EADh) PERF_UNIT_CTL_CHA_14 on page 407
EAeh	(EAeh) PERF_UNIT_CTL_1_CHA_14 on page 408
EAFh	(EAFh) PERF_UNIT_STATUS_CHA_14 on page 408
EB0h	(EB0h) PERF_CTR_0_CHA_14 on page 409
EB1h	(EB1h) PERF_CTR_1_CHA_14 on page 409
EB2h	(EB2h) PERF_CTR_2_CHA_14 on page 409
EB3h	(EB3h) PERF_CTR_3_CHA_14 on page 409
EB4h	(EB4h) PERF_CTR_UNIT_CTRL_CHA_15 on page 410
EB5h	(EB5h) PERF_EVT_SEL_0_CHA_15 on page 410
EB6h	(EB6h) PERF_EVT_SEL_1_CHA_15 on page 411
EB7h	(EB7h) PERF_EVT_SEL_2_CHA_15 on page 413
EB8h	(EB8h) PERF_EVT_SEL_3_CHA_15 on page 414
EB9h	(EB9h) PERF_UNIT_CTL_CHA_15 on page 415
EBAh	(EBAh) PERF_UNIT_CTL_1_CHA_15 on page 416
EBBh	(EBBh) PERF_UNIT_STATUS_CHA_15 on page 417
EBCh	(EBCh) PERF_CTR_0_CHA_15 on page 417
EBDh	(EBDh) PERF_CTR_1_CHA_15 on page 417
EBEh	(EBEh) PERF_CTR_2_CHA_15 on page 417
EBFh	(EBFh) PERF_CTR_3_CHA_15 on page 417
EC0h	(EC0h) PERF_CTR_UNIT_CTRL_CHA_16 on page 418
EC1h	(EC1h) PERF_EVT_SEL_0_CHA_16 on page 418
continued...	



Offset Start	Register ID
EC2h	(EC2h) PERF_EVT_SEL_1_CHA_16 on page 420
EC3h	(EC3h) PERF_EVT_SEL_2_CHA_16 on page 421
EC4h	(EC4h) PERF_EVT_SEL_3_CHA_16 on page 422
EC5h	(EC5h) PERF_UNIT_CTL_CHA_16 on page 424
EC6h	(EC6h) PERF_UNIT_CTL_1_CHA_16 on page 424
EC7h	(EC7h) PERF_UNIT_STATUS_CHA_16 on page 425
EC8h	(EC8h) PERF_CTR_0_CHA_16 on page 425
EC9h	(EC9h) PERF_CTR_1_CHA_16 on page 425
ECAh	(ECAh) PERF_CTR_2_CHA_16 on page 426
ECBh	(ECBh) PERF_CTR_3_CHA_16 on page 426
ECCh	(ECCh) PERF_CTR_UNIT_CTRL_CHA_17 on page 426
ECDh	(ECDh) PERF_EVT_SEL_0_CHA_17 on page 427
ECEh	(ECEh) PERF_EVT_SEL_1_CHA_17 on page 428
ECFh	(ECFh) PERF_EVT_SEL_2_CHA_17 on page 429
ED0h	(ED0h) PERF_EVT_SEL_3_CHA_17 on page 431
ED1h	(ED1h) PERF_UNIT_CTL_CHA_17 on page 432
ED2h	(ED2h) PERF_UNIT_CTL_1_CHA_17 on page 432
ED3h	(ED3h) PERF_UNIT_STATUS_CHA_17 on page 433
ED4h	(ED4h) PERF_CTR_0_CHA_17 on page 433
ED5h	(ED5h) PERF_CTR_1_CHA_17 on page 434
ED6h	(ED6h) PERF_CTR_2_CHA_17 on page 434
ED7h	(ED7h) PERF_CTR_3_CHA_17 on page 434
ED8h	(ED8h) PERF_CTR_UNIT_CTRL_CHA_18 on page 434
ED9h	(ED9h) PERF_EVT_SEL_0_CHA_18 on page 435
EDAh	(EDAh) PERF_EVT_SEL_1_CHA_18 on page 436
EDBh	(EDBh) PERF_EVT_SEL_2_CHA_18 on page 437
EDCh	(EDCh) PERF_EVT_SEL_3_CHA_18 on page 439
EDDh	(EDDh) PERF_UNIT_CTL_CHA_18 on page 440
EDEh	(EDEh) PERF_UNIT_CTL_1_CHA_18 on page 441
EDFh	(EDFh) PERF_UNIT_STATUS_CHA_18 on page 441
EE0h	(EE0h) PERF_CTR_0_CHA_18 on page 442
EE1h	(EE1h) PERF_CTR_1_CHA_18 on page 442
EE2h	(EE2h) PERF_CTR_2_CHA_18 on page 442
EE3h	(EE3h) PERF_CTR_3_CHA_18 on page 442
EE4h	(EE4h) PERF_CTR_UNIT_CTRL_CHA_19 on page 443
EE5h	(EE5h) PERF_EVT_SEL_0_CHA_19 on page 443
continued...	



Offset Start	Register ID
EE6h	(EE6h) PERF_EVT_SEL_1_CHA_19 on page 444
EE7h	(EE7h) PERF_EVT_SEL_2_CHA_19 on page 446
EE8h	(EE8h) PERF_EVT_SEL_3_CHA_19 on page 447
EE9h	(EE9h) PERF_UNIT_CTL_CHA_19 on page 448
EEAh	(EEAh) PERF_UNIT_CTL_1_CHA_19 on page 449
EEBh	(EEBh) PERF_UNIT_STATUS_CHA_19 on page 450
EECh	(EECh) PERF_CTR_0_CHA_19 on page 450
EEDh	(EEDh) PERF_CTR_1_CHA_19 on page 450
EEEnh	(EEEnh) PERF_CTR_2_CHA_19 on page 450
EEFh	(EEFh) PERF_CTR_3_CHA_19 on page 450
EF0h	(EF0h) PERF_CTR_UNIT_CTRL_CHA_20 on page 451
EF1h	(EF1h) PERF_EVT_SEL_0_CHA_20 on page 451
EF2h	(EF2h) PERF_EVT_SEL_1_CHA_20 on page 453
EF3h	(EF3h) PERF_EVT_SEL_2_CHA_20 on page 454
EF4h	(EF4h) PERF_EVT_SEL_3_CHA_20 on page 455
EF5h	(EF5h) PERF_UNIT_CTL_CHA_20 on page 457
EF6h	(EF6h) PERF_UNIT_CTL_1_CHA_20 on page 457
EF7h	(EF7h) PERF_UNIT_STATUS_CHA_20 on page 458
EF8h	(EF8h) PERF_CTR_0_CHA_20 on page 458
EF9h	(EF9h) PERF_CTR_1_CHA_20 on page 458
EFAh	(EFAh) PERF_CTR_2_CHA_20 on page 459
EFBh	(EFBh) PERF_CTR_3_CHA_20 on page 459
EFCh	(EFCh) PERF_CTR_UNIT_CTRL_CHA_21 on page 459
EFDh	(EFDh) PERF_EVT_SEL_0_CHA_21 on page 460
EFEh	(EFEh) PERF_EVT_SEL_1_CHA_21 on page 461
EFFh	(EFFh) PERF_EVT_SEL_2_CHA_21 on page 462
F00h	(F00h) PERF_EVT_SEL_3_CHA_21 on page 464
F01h	(F01h) PERF_UNIT_CTL_CHA_21 on page 465
F02h	(F02h) PERF_UNIT_CTL_1_CHA_21 on page 465
F03h	(F03h) PERF_UNIT_STATUS_CHA_21 on page 466
F04h	(F04h) PERF_CTR_0_CHA_21 on page 466
F05h	(F05h) PERF_CTR_1_CHA_21 on page 467
F06h	(F06h) PERF_CTR_2_CHA_21 on page 467
F07h	(F07h) PERF_CTR_3_CHA_21 on page 467
F08h	(F08h) PERF_CTR_UNIT_CTRL_CHA_22 on page 467
F09h	(F09h) PERF_EVT_SEL_0_CHA_22 on page 468
continued...	



Offset Start	Register ID
F0Ah	(F0Ah) PERF_EVT_SEL_1_CHA_22 on page 469
F0Bh	(F0Bh) PERF_EVT_SEL_2_CHA_22 on page 470
F0Ch	(F0Ch) PERF_EVT_SEL_3_CHA_22 on page 472
F0Dh	(F0Dh) PERF_UNIT_CTL_CHA_22 on page 473
F0Eh	(F0Eh) PERF_UNIT_CTL_1_CHA_22 on page 474
F0Fh	(F0Fh) PERF_UNIT_STATUS_CHA_22 on page 474
F10h	(F10h) PERF_CTR_0_CHA_22 on page 475
F11h	(F11h) PERF_CTR_1_CHA_22 on page 475
F12h	(F12h) PERF_CTR_2_CHA_22 on page 475
F13h	(F13h) PERF_CTR_3_CHA_22 on page 475
F14h	(F14h) PERF_CTR_UNIT_CTRL_CHA_23 on page 476
F15h	(F15h) PERF_EVT_SEL_0_CHA_23 on page 476
F16h	(F16h) PERF_EVT_SEL_1_CHA_23 on page 477
F17h	(F17h) PERF_EVT_SEL_2_CHA_23 on page 479
F18h	(F18h) PERF_EVT_SEL_3_CHA_23 on page 480
F19h	(F19h) PERF_UNIT_CTL_CHA_23 on page 481
F1Ah	(F1Ah) PERF_UNIT_CTL_1_CHA_23 on page 482
F1Bh	(F1Bh) PERF_UNIT_STATUS_CHA_23 on page 483
F1Ch	(F1Ch) PERF_CTR_0_CHA_23 on page 483
F1Dh	(F1Dh) PERF_CTR_1_CHA_23 on page 483
F1Eh	(F1Eh) PERF_CTR_2_CHA_23 on page 483
F1Fh	(F1Fh) PERF_CTR_3_CHA_23 on page 483
F20h	(F20h) PERF_CTR_UNIT_CTRL_CHA_24 on page 484
F21h	(F21h) PERF_EVT_SEL_0_CHA_24 on page 484
F22h	(F22h) PERF_EVT_SEL_1_CHA_24 on page 486
F23h	(F23h) PERF_EVT_SEL_2_CHA_24 on page 487
F24h	(F24h) PERF_EVT_SEL_3_CHA_24 on page 488
F25h	(F25h) PERF_UNIT_CTL_CHA_24 on page 490
F26h	(F26h) PERF_UNIT_CTL_1_CHA_24 on page 490
F27h	(F27h) PERF_UNIT_STATUS_CHA_24 on page 491
F28h	(F28h) PERF_CTR_0_CHA_24 on page 491
F29h	(F29h) PERF_CTR_1_CHA_24 on page 491
F2Ah	(F2Ah) PERF_CTR_2_CHA_24 on page 492
F2Bh	(F2Bh) PERF_CTR_3_CHA_24 on page 492
F2Ch	(F2Ch) PERF_CTR_UNIT_CTRL_CHA_25 on page 492
F2Dh	(F2Dh) PERF_EVT_SEL_0_CHA_25 on page 493
continued...	



Offset Start	Register ID
F2Eh	(F2Eh) PERF_EVT_SEL_1_CHA_25 on page 494
F2Fh	(F2Fh) PERF_EVT_SEL_2_CHA_25 on page 495
F30h	(F30h) PERF_EVT_SEL_3_CHA_25 on page 497
F31h	(F31h) PERF_UNIT_CTL_CHA_25 on page 498
F32h	(F32h) PERF_UNIT_CTL_1_CHA_25 on page 498
F33h	(F33h) PERF_UNIT_STATUS_CHA_25 on page 499
F34h	(F34h) PERF_CTR_0_CHA_25 on page 499
F35h	(F35h) PERF_CTR_1_CHA_25 on page 500
F36h	(F36h) PERF_CTR_2_CHA_25 on page 500
F37h	(F37h) PERF_CTR_3_CHA_25 on page 500
F38h	(F38h) PERF_CTR_UNIT_CTRL_CHA_26 on page 500
F39h	(F39h) PERF_EVT_SEL_0_CHA_26 on page 501
F3Ah	(F3Ah) PERF_EVT_SEL_1_CHA_26 on page 502
F3Bh	(F3Bh) PERF_EVT_SEL_2_CHA_26 on page 503
F3Ch	(F3Ch) PERF_EVT_SEL_3_CHA_26 on page 505
F3Dh	(F3Dh) PERF_UNIT_CTL_CHA_26 on page 506
F3Eh	(F3Eh) PERF_UNIT_CTL_1_CHA_26 on page 507
F3Fh	(F3Fh) PERF_UNIT_STATUS_CHA_26 on page 507
F40h	(F40h) PERF_CTR_0_CHA_26 on page 508
F41h	(F41h) PERF_CTR_1_CHA_26 on page 508
F42h	(F42h) PERF_CTR_2_CHA_26 on page 508
F43h	(F43h) PERF_CTR_3_CHA_26 on page 508
F44h	(F44h) PERF_CTR_UNIT_CTRL_CHA_27 on page 509
F45h	(F45h) PERF_EVT_SEL_0_CHA_27 on page 509
F46h	(F46h) PERF_EVT_SEL_1_CHA_27 on page 510
F47h	(F47h) PERF_EVT_SEL_2_CHA_27 on page 512
F48h	(F48h) PERF_EVT_SEL_3_CHA_27 on page 513
F49h	(F49h) PERF_UNIT_CTL_CHA_27 on page 514
F4Ah	(F4Ah) PERF_UNIT_CTL_1_CHA_27 on page 515
F4Bh	(F4Bh) PERF_UNIT_STATUS_CHA_27 on page 516
F4Ch	(F4Ch) PERF_CTR_0_CHA_27 on page 516
F4Dh	(F4Dh) PERF_CTR_1_CHA_27 on page 516
F4Eh	(F4Eh) PERF_CTR_2_CHA_27 on page 516
F4Fh	(F4Fh) PERF_CTR_3_CHA_27 on page 516
F50h	(F50h) PERF_CTR_UNIT_CTRL_CHA_28 on page 517
F51h	(F51h) PERF_EVT_SEL_0_CHA_28 on page 517
continued...	



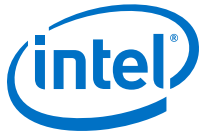
Offset Start	Register ID
F52h	(F52h) PERF_EVT_SEL_1_CHA_28 on page 519
F53h	(F53h) PERF_EVT_SEL_2_CHA_28 on page 520
F54h	(F54h) PERF_EVT_SEL_3_CHA_28 on page 521
F55h	(F55h) PERF_UNIT_CTL_CHA_28 on page 523
F56h	(F56h) PERF_UNIT_CTL_1_CHA_28 on page 523
F57h	(F57h) PERF_UNIT_STATUS_CHA_28 on page 524
F58h	(F58h) PERF_CTR_0_CHA_28 on page 524
F59h	(F59h) PERF_CTR_1_CHA_28 on page 524
F5Ah	(F5Ah) PERF_CTR_2_CHA_28 on page 525
F5Bh	(F5Bh) PERF_CTR_3_CHA_28 on page 525
F5Ch	(F5Ch) PERF_CTR_UNIT_CTRL_CHA_29 on page 525
F5Dh	(F5Dh) PERF_EVT_SEL_0_CHA_29 on page 526
F5Eh	(F5Eh) PERF_EVT_SEL_1_CHA_29 on page 527
F5Fh	(F5Fh) PERF_EVT_SEL_2_CHA_29 on page 528
F60h	(F60h) PERF_EVT_SEL_3_CHA_29 on page 530
F61h	(F61h) PERF_UNIT_CTL_CHA_29 on page 531
F62h	(F62h) PERF_UNIT_CTL_1_CHA_29 on page 531
F63h	(F63h) PERF_UNIT_STATUS_CHA_29 on page 532
F64h	(F64h) PERF_CTR_0_CHA_29 on page 532
F65h	(F65h) PERF_CTR_1_CHA_29 on page 533
F66h	(F66h) PERF_CTR_2_CHA_29 on page 533
F67h	(F67h) PERF_CTR_3_CHA_29 on page 533
F68h	(F68h) PERF_CTR_UNIT_CTRL_CHA_30 on page 533
F69h	(F69h) PERF_EVT_SEL_0_CHA_30 on page 534
F6Ah	(F6Ah) PERF_EVT_SEL_1_CHA_30 on page 535
F6Bh	(F6Bh) PERF_EVT_SEL_2_CHA_30 on page 536
F6Ch	(F6Ch) PERF_EVT_SEL_3_CHA_30 on page 538
F6Dh	(F6Dh) PERF_UNIT_CTL_CHA_30 on page 539
F6Eh	(F6Eh) PERF_UNIT_CTL_1_CHA_30 on page 540
F6Fh	(F6Fh) PERF_UNIT_STATUS_CHA_30 on page 540
F70h	(F70h) PERF_CTR_0_CHA_30 on page 541
F71h	(F71h) PERF_CTR_1_CHA_30 on page 541
F72h	(F72h) PERF_CTR_2_CHA_30 on page 541
F73h	(F73h) PERF_CTR_3_CHA_30 on page 541
F74h	(F74h) PERF_CTR_UNIT_CTRL_CHA_31 on page 542
F75h	(F75h) PERF_EVT_SEL_0_CHA_31 on page 542
continued...	



Offset Start	Register ID
F76h	(F76h) PERF_EVT_SEL_1_CHA_31 on page 543
F77h	(F77h) PERF_EVT_SEL_2_CHA_31 on page 545
F78h	(F78h) PERF_EVT_SEL_3_CHA_31 on page 546
F79h	(F79h) PERF_UNIT_CTL_CHA_31 on page 547
F7Ah	(F7Ah) PERF_UNIT_CTL_1_CHA_31 on page 548
F7Bh	(F7Bh) PERF_UNIT_STATUS_CHA_31 on page 549
F7Ch	(F7Ch) PERF_CTR_0_CHA_31 on page 549
F7Dh	(F7Dh) PERF_CTR_1_CHA_31 on page 549
F7Eh	(F7Eh) PERF_CTR_2_CHA_31 on page 549
F7Fh	(F7Fh) PERF_CTR_3_CHA_31 on page 549
F80h	(F80h) PERF_CTR_UNIT_CTRL_CHA_32 on page 550
F81h	(F81h) PERF_EVT_SEL_0_CHA_32 on page 550
F82h	(F82h) PERF_EVT_SEL_1_CHA_32 on page 552
F83h	(F83h) PERF_EVT_SEL_2_CHA_32 on page 553
F84h	(F84h) PERF_EVT_SEL_3_CHA_32 on page 554
F85h	(F85h) PERF_UNIT_CTL_CHA_32 on page 556
F86h	(F86h) PERF_UNIT_CTL_1_CHA_32 on page 556
F87h	(F87h) PERF_UNIT_STATUS_CHA_32 on page 557
F88h	(F88h) PERF_CTR_0_CHA_32 on page 557
F89h	(F89h) PERF_CTR_1_CHA_32 on page 557
F8Ah	(F8Ah) PERF_CTR_2_CHA_32 on page 558
F8Bh	(F8Bh) PERF_CTR_3_CHA_32 on page 558
F8Ch	(F8Ch) PERF_CTR_UNIT_CTRL_CHA_33 on page 558
F8Dh	(F8Dh) PERF_EVT_SEL_0_CHA_33 on page 559
F8Eh	(F8Eh) PERF_EVT_SEL_1_CHA_33 on page 560
F8Fh	(F8Fh) PERF_EVT_SEL_2_CHA_33 on page 561
F90h	(F90h) PERF_EVT_SEL_3_CHA_33 on page 563
F91h	(F91h) PERF_UNIT_CTL_CHA_33 on page 564
F92h	(F92h) PERF_UNIT_CTL_1_CHA_33 on page 564
F93h	(F93h) PERF_UNIT_STATUS_CHA_33 on page 565
F94h	(F94h) PERF_CTR_0_CHA_33 on page 565
F95h	(F95h) PERF_CTR_1_CHA_33 on page 566
F96h	(F96h) PERF_CTR_2_CHA_33 on page 566
F97h	(F97h) PERF_CTR_3_CHA_33 on page 566
F98h	(F98h) PERF_CTR_UNIT_CTRL_CHA_34 on page 566
F99h	(F99h) PERF_EVT_SEL_0_CHA_34 on page 567
continued...	



Offset Start	Register ID
F9Ah	(F9Ah) PERF_EVT_SEL_1_CHA_34 on page 568
F9Bh	(F9Bh) PERF_EVT_SEL_2_CHA_34 on page 569
F9Ch	(F9Ch) PERF_EVT_SEL_3_CHA_34 on page 571
F9Dh	(F9Dh) PERF_UNIT_CTL_CHA_34 on page 572
F9Eh	(F9Eh) PERF_UNIT_CTL_1_CHA_34 on page 573
F9Fh	(F9Fh) PERF_UNIT_STATUS_CHA_34 on page 573
FA0h	(FA0h) PERF_CTR_0_CHA_34 on page 574
FA1h	(FA1h) PERF_CTR_1_CHA_34 on page 574
FA2h	(FA2h) PERF_CTR_2_CHA_34 on page 574
FA3h	(FA3h) PERF_CTR_3_CHA_34 on page 574
FA4h	(FA4h) PERF_CTR_UNIT_CTRL_CHA_35 on page 575
FA5h	(FA5h) PERF_EVT_SEL_0_CHA_35 on page 575
FA6h	(FA6h) PERF_EVT_SEL_1_CHA_35 on page 576
FA7h	(FA7h) PERF_EVT_SEL_2_CHA_35 on page 578
FA8h	(FA8h) PERF_EVT_SEL_3_CHA_35 on page 579
FA9h	(FA9h) PERF_UNIT_CTL_CHA_35 on page 580
FAAh	(FAAh) PERF_UNIT_CTL_1_CHA_35 on page 581
FABh	(FABh) PERF_UNIT_STATUS_CHA_35 on page 582
FACh	(FACh) PERF_CTR_0_CHA_35 on page 582
FADh	(FADh) PERF_CTR_1_CHA_35 on page 582
FAEh	(FAEh) PERF_CTR_2_CHA_35 on page 582
FAFh	(FAFh) PERF_CTR_3_CHA_35 on page 582
FB0h	(FB0h) PERF_CTR_UNIT_CTRL_CHA_36 on page 583
FB1h	(FB1h) PERF_EVT_SEL_0_CHA_36 on page 583
FB2h	(FB2h) PERF_EVT_SEL_1_CHA_36 on page 585
FB3h	(FB3h) PERF_EVT_SEL_2_CHA_36 on page 586
FB4h	(FB4h) PERF_EVT_SEL_3_CHA_36 on page 587
FB5h	(FB5h) PERF_UNIT_CTL_CHA_36 on page 589
FB6h	(FB6h) PERF_UNIT_CTL_1_CHA_36 on page 589
FB7h	(FB7h) PERF_UNIT_STATUS_CHA_36 on page 590
FB8h	(FB8h) PERF_CTR_0_CHA_36 on page 590
FB9h	(FB9h) PERF_CTR_1_CHA_36 on page 590
FBAh	(FBAh) PERF_CTR_2_CHA_36 on page 591
FBBh	(FBBh) PERF_CTR_3_CHA_36 on page 591
FBCh	(FBCh) PERF_CTR_UNIT_CTRL_CHA_37 on page 591
FBDh	(FBDh) PERF_EVT_SEL_0_CHA_37 on page 592
continued...	



Offset Start	Register ID
FBEh	(FBEh) PERF_EVT_SEL_1_CHA_37 on page 593
FBFh	(FBFh) PERF_EVT_SEL_2_CHA_37 on page 594
FC0h	(FC0h) PERF_EVT_SEL_3_CHA_37 on page 596
FC1h	(FC1h) PERF_UNIT_CTL_CHA_37 on page 597
FC2h	(FC2h) PERF_UNIT_CTL_1_CHA_37 on page 597
FC3h	(FC3h) PERF_UNIT_STATUS_CHA_37 on page 598
FC4h	(FC4h) PERF_CTR_0_CHA_37 on page 598
FC5h	(FC5h) PERF_CTR_1_CHA_37 on page 599
FC6h	(FC6h) PERF_CTR_2_CHA_37 on page 599
FC7h	(FC7h) PERF_CTR_3_CHA_37 on page 599

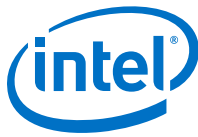
5.2.1 (700h) NcuPMONGICtrl

Global PMON All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCtrClr] is set to 1b.

MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
63	package	0h	WO	Freeze Counter (FrzCounter) — FreezCounter : This bit, when asserted, cause all of the global enables for the UnCore PerfMon Counters and fixed counters to be frozen. 0 : No effect 1 : Freeze all the perfmon counters, stop counting
62	package	0h	RW_V	Wake on PMI (WkOnPMI) — This bit determines whether PMI event is sent to waken cores only or is broadcast to all cores after waking up any sleeping core. 0 : Avoid waking a core for PMI event - send event to waken cores only. 1 : Wake any sleeping core and send PMI event to all cores
61	package	0h	WO	PMON Unfreeze Counters (UnfrzCounter) — UnfreezCounter : This bit, when asserted, cause all of the global enables for the UnCore PerfMon Counters and fixed counters to be frozen. 0 : No effect 1 : Unfreeze all the perfmon counters, start counting
60	package	0h	RW_V	PMI Overflow Enable uBP (PMIOvfEnUBP) — Enable the generation of uBP event when set. 0 - Does not assert uBP event with the PMI Overflow. 1 - Assert uBP event with the PMI Overflow is set.
59	package	0h	WO	Reset All Perfmon Counters (ResetCnts) — ResetCounter : This bit, when asserted, cause all of the global enables for the UnCore PerfMon Counters and fixed counters to be frozen. 0 : No effect 1 : Reset all the perfmon counters
58:38	-	-	-	Reserved (RSVD) — Reserved.
37	package	0h	RW_V	PMI Select to Core 37 (PMISelC37) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI
continued...				



MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
				event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
36	package	0h	RW_V	PMI Select to Core 36 (PMISelC36) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
35	package	0h	RW_V	PMI Select to Core 35 (PMISelC35) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
34	package	0h	RW_V	PMI Select to Core 34 (PMISelC34) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
33	package	0h	RW_V	PMI Select to Core 33 (PMISelC33) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
32	package	0h	RW_V	PMI Select to Core 32 (PMISelC32) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
31	package	0h	RW_V	PMI Select to Core 31 (PMISelC31) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
30	package	0h	RW_V	PMI Select to Core 30 (PMISelC30) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
29	package	0h	RW_V	PMI Select to Core 29 (PMISelC29) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise,
continued...				



MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
				the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
28	package	0h	RW_V	PMI Select to Core 28 (PMISelC28) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
27	package	0h	RW_V	PMI Select to Core 27 (PMISelC27) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
26	package	0h	RW_V	PMI Select to Core 26 (PMISelC26) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
25	package	0h	RW_V	PMI Select to Core 25 (PMISelC25) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
24	package	0h	RW_V	PMI Select to Core 24 (PMISelC24) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
23	package	0h	RW_V	PMI Select to Core 23 (PMISelC23) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
22	package	0h	RW_V	PMI Select to Core 22 (PMISelC22) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
continued...				



MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RW_V	PMI Select to Core 21 (PMISelC21) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
20	package	0h	RW_V	PMI Select to Core 20 (PMISelC20) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
19	package	0h	RW_V	PMI Select to Core 19 (PMISelC19) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
18	package	0h	RW_V	PMI Select to Core 18 (PMISelC18) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
17	package	0h	RW_V	PMI Select to Core 17 (PMISelC17) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
16	package	0h	RW_V	PMI Select to Core 16 (PMISelC16) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
15	package	0h	RW_V	PMI Select to Core 15 (PMISelC15) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
14	package	0h	RW_V	PMI Select to Core 14 (PMISelC14) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
continued...				



MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
13	package	0h	RW_V	PMI Select to Core 13 (PMISelC13) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
12	package	0h	RW_V	PMI Select to Core 12 (PMISelC12) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
11	package	0h	RW_V	PMI Select to Core 11 (PMISelC11) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
10	package	0h	RW_V	PMI Select to Core 10 (PMISelC10) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
9	package	0h	RW_V	PMI Select to Core 9 (PMISelC9) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
8	package	0h	RW_V	PMI Select to Core 8 (PMISelC8) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
7	package	0h	RW_V	PMI Select to Core 7 (PMISelC7) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
6	package	0h	RW_V	PMI Select to Core 6 (PMISelC6) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
continued...				



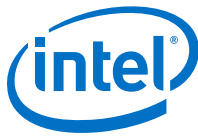
MSR Address: 700h				
Bit	Scope	Default	Attribute	Description
5	package	0h	RW_V	PMI Select to Core 5 (PMISelC5) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
4	package	0h	RW_V	PMI Select to Core 4 (PMISelC4) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
3	package	0h	RW_V	PMI Select to Core 3 (PMISelC3) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
2	package	0h	RW_V	PMI Select to Core 2 (PMISelC2) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
1	package	0h	RW_V	PMI Select to Core 1 (PMISelC1) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core
0	package	0h	RW_V	PMI Select to Core 0 (PMISelC0) — Enables sending the PMI to the respective core. If WakeOnPMI is '1' a wake request is sent to any sleeping core within the mask prior to sending the event. Otherwise, the PMON event is sent to all waken cores within the mask. The PMI event is sent as a set of un-cast messages to the target cores. 0 - Don't send event to the core (default) 1 - Send event to the core

5.2.2 (701h) NcuPMONGISTs

PMON Global Status register. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCntClr] is set to 1b.

MSR Address: 701h				
Bit	Scope	Default	Attribute	Description
63:53	-	-	-	Reserved (RSVD) — Reserved.
52	package	0h	RW1C	IIO Overflow (IIOOvf) — Set by hardware when a counter overflow occurs in IIO(CBDMA)

continued...



MSR Address: 701h				
Bit	Scope	Default	Attribute	Description
51	package	0h	RW1C	M2PCIe Overflow (MSPCIeOvf) — Set by hardware when a counter overflow occurs in M2PCIe
50	package	0h	RW1C	Overflow in the EDC 7 (EDC7Ovf) — Set by hardware when an overflow occurs in the EDC 7
49	package	0h	RW1C	Overflow in the EDC 6 (EDC6Ovf) — Set by hardware when an overflow occurs in the EDC 6
48	package	0h	RW1C	Overflow in the EDC 5 (EDC5Ovf) — Set by hardware when an overflow occurs in the EDC 5
47	package	0h	RW1C	Overflow in the EDC 4 (EDC4Ovf) — Set by hardware when an overflow occurs in the EDC 4
46	package	0h	RW1C	Overflow in the EDC 3 (EDC3Ovf) — Set by hardware when an overflow occurs in the EDC 3
45	package	0h	RW1C	Overflow in the EDC 2 (EDC2Ovf) — Set by hardware when an overflow occurs in the EDC 2
44	package	0h	RW1C	Overflow in the EDC 1 (EDC1Ovf) — Set by hardware when an overflow occurs in the EDC 1
43	package	0h	RW1C	Overflow in the EDC 0 (EDC0Ovf) — Set by hardware when an overflow occurs in the EDC 0
42	package	0h	RW1C	Overflow in the memory controller 1 (iMC1Ovf) — Set by hardware when an overflow occurs in the memory controller 1
41	package	0h	RW1C	Overflow in the memory controller 0 (iMC0Ovf) — Set by hardware when an overflow occurs in the memory controller 0
40	package	0h	RW1C	CBo37 Overflow (CBo37Ovf) — Set by hardware when an overflow occurs in CBo 37
39	package	0h	RW1C	CBo36 Overflow (CBo36Ovf) — Set by hardware when an overflow occurs in CBo 36
38	package	0h	RW1C	CBo35 Overflow (CBo35Ovf) — Set by hardware when an overflow occurs in CBo 35
37	package	0h	RW1C	CBo34 Overflow (CBo34Ovf) — Set by hardware when an overflow occurs in CBo 34
36	package	0h	RW1C	CBo33 Overflow (CBo33Ovf) — Set by hardware when an overflow occurs in CBo 33
35	package	0h	RW1C	CBo32 Overflow (CBo32Ovf) — Set by hardware when an overflow occurs in CBo 32
34	package	0h	RW1C	CBo31 Overflow (CBo31Ovf) — Set by hardware when an overflow occurs in CBo 31
33	package	0h	RW1C	CBo30 Overflow (CBo30Ovf) — Set by hardware when an overflow occurs in CBo 30
32	package	0h	RW1C	CBo29 Overflow (CBo29Ovf) — Set by hardware when an overflow occurs in CBo 29
31	package	0h	RW1C	CBo28 Overflow (CBo28Ovf) — Set by hardware when an overflow occurs in CBo 28
30	package	0h	RW1C	CBo27 Overflow (CBo27Ovf) — Set by hardware when an overflow occurs in CBo 27
29	package	0h	RW1C	CBo26 Overflow (CBo26Ovf) — Set by hardware when an overflow occurs in CBo 26
continued...				



MSR Address: 701h				
Bit	Scope	Default	Attribute	Description
28	package	0h	RW1C	CBO25 Overflow (CBo25Ovf) — Set by hardware when an overflow occurs in CBo 25
27	package	0h	RW1C	CBO24 Overflow (CBo24Ovf) — Set by hardware when an overflow occurs in CBo 24
26	package	0h	RW1C	CBO23 Overflow (CBo23Ovf) — Set by hardware when an overflow occurs in CBo 23
25	package	0h	RW1C	CBO22 Overflow (CBo22Ovf) — Set by hardware when an overflow occurs in CBo 22
24	package	0h	RW1C	CBO21 Overflow (CBo21Ovf) — Set by hardware when an overflow occurs in CBo 21
23	package	0h	RW1C	CBO20 Overflow (CBo20Ovf) — Set by hardware when an overflow occurs in CBo 20
22	package	0h	RW1C	CBO19 Overflow (CBo19Ovf) — Set by hardware when an overflow occurs in CBo 19
21	package	0h	RW1C	CBO18 Overflow (CBo18Ovf) — Set by hardware when an overflow occurs in CBo 18
20	package	0h	RW1C	CBO17 Overflow (CBo17Ovf) — Set by hardware when an overflow occurs in CBo 17
19	package	0h	RW1C	CBO16 Overflow (CBo16Ovf) — Set by hardware when an overflow occurs in CBo 17
18	package	0h	RW1C	CBO15 Overflow (CBo15Ovf) — Set by hardware when an overflow occurs in CBo 17
17	package	0h	RW1C	CBO14 Overflow (CBo14Ovf) — Set by hardware when an overflow occurs in CBo 17
16	package	0h	RW1C	CBO13 Overflow (CBo13Ovf) — Set by hardware when an overflow occurs in CBo 16
15	package	0h	RW1C	CBO12 Overflow (CBo12Ovf) — Set by hardware when an overflow occurs in CBo 15
14	package	0h	RW1C	CBO11 Overflow (CBo11Ovf) — Set by hardware when an overflow occurs in CBo 11
13	package	0h	RW1C	CBO10 Overflow (CBo10Ovf) — Set by hardware when an overflow occurs in CBo 10
12	package	0h	RW1C	CBO9 Overflow (CBo9Ovf) — Set by hardware when an overflow occurs in CBo 9
11	package	0h	RW1C	CBO8 Overflow (CBo8Ovf) — Set by hardware when an overflow occurs in CBo 8
10	package	0h	RW1C	CBO7 Overflow (CBo7Ovf) — Set by hardware when an overflow occurs in CBo 7
9	package	0h	RW1C	CBO6 Overflow (CBo6Ovf) — Set by hardware when an overflow occurs in CBo 6
8	package	0h	RW1C	CBO5 Overflow (CBo5Ovf) — Set by hardware when an overflow occurs in CBo 5
7	package	0h	RW1C	CBO4 Overflow (CBo4Ovf) — Set by hardware when an overflow occurs in CBo 4
6	package	0h	RW1C	CBO3 Overflow (CBo3Ovf) — Set by hardware when an overflow occurs in CBo 3
continued...				



MSR Address: 701h				
Bit	Scope	Default	Attribute	Description
5	package	0h	RW1C	CBO2 Overflow (CBo2Ovf) — Set by hardware when an overflow occurs in CBo 2
4	package	0h	RW1C	CBO1 Overflow (CBo1Ovf) — Set by hardware when an overflow occurs in CBo 1
3	package	0h	RW1C	CBO0 Overflow (CBo0Ovf) — Set by hardware when an overflow occurs in CBo 0
2	package	0h	RW1C	PCU Overflow (PCUOvf) — Set by hardware to indicate that an overflow happened in the PCU
1	package	0h	RW1C	NCU Counters Overflow (NcuCtrOvf) — Indicates that an overflow event occurred on the NCU and IMPH Event. read 0 - No overflow detected 1 - Event overflow event was detected (sticky) Write 0 - Ignore 1 - Clear this bit
0	package	0h	RW1C	Fixed Counter Overflow (FixCtrOvf) — Indicates that an overflow event occurred on the Fix Event. read 0 - No overflow detected 1 - Event overflow event was detected (sticky) Write 0 - Ignore 1 - Clear this bit

5.2.3 (702h) NCUPMONConfig

This is MSR that shows to user configuration of uncore PMON's. In this particular case it shows the number of Cbo PMON banks

MSR Address: 702h				
Bit	Scope	Default	Attribute	Description
31:6	-	-	-	Reserved (RSVD) — Reserved.
5:0	package	8h	RW	Number of Cbo PMON set's (NumofCboPMON) — Reserved.

5.2.4 (703h) NcuPMONCtlFix

Holds the status information and control the operation of the PMON Fixed Counter. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCtrClr] is set to 1b.

MSR Address: 703h				
Bit	Scope	Default	Attribute	Description
31:23	-	-	-	Reserved (RSVD) — Reserved.
22	package	0h	RW_V	Counter Enable (CounterEn) — Enable the counter to count when the global enable is active. 0 : counter is disabled and will not count when global enable is set. 1 : counter is enabled and will count when global enable is set.
continued...				



MSR Address: 703h				
Bit	Scope	Default	Attribute	Description
21	-	-	-	Reserved (RSVD) — Reserved.
20	package	0h	RW_V	Overflow Enable (OvfEn) — Enable generation of Overflow indication when this counter overflows. 0 : PMI Event for this counter is disabled 1 : PMI event is enabled
19:0	-	-	-	Reserved (RSVD) — Reserved.

5.2.5 (704h) NcuPMONCtrFx

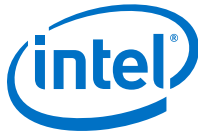
PMON Fixed Counter data register The Fix counter is counting UCLK cycles. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCntClr] is set to 1b.

MSR Address: 704h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	PMON Counter Data (PmonCtrData) — Pmon Counter data. The register may be read and written when the counter is stopped.

5.2.6 (705h) NcuPMONCtCtl1

Controls the operation of the the PMON Counter 1. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCtrClr] is set to 1b.

MSR Address: 705h				
Bit	Scope	Default	Attribute	Description
31:29	-	-	-	Reserved (RSVD) — Reserved.
28:24	package	0h	RW	Threshold — This field is compared directly against an incoming increment value from event increment bus. The width of this field is determined by the widest event in the iMPH (which is 5 bits for queue occupancy). The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the Invert bit) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification (i.e. the counter will advance by the specified increment value).
23	package	0h	RW	Intert Event (Invert) — Invert the event for counting purposes : 0 : Count when the event is asserted or on it's rising edge 1. : Count when the event is de-asserted or on it's falling edge
22	package	0h	RW	Counter Enable (CounterEn) — Enable the counter to count when the global enable is active. 0 : counter is disabled and will not count when global enable is set. 1 : counter is enabled and will count when global enable is set.
21	-	-	-	Reserved (RSVD) — Reserved.
20	package	0h	RW	Overflow Enable (OvfEn) — Enable generation of Overflow indication when this counter overflows. 0 : PMI Event for this counter is disabled 1 : PMI event is enabled
continued...				



MSR Address: 705h				
Bit	Scope	Default	Attribute	Description
19	package	0h	RW	TID filter enable (TidFilterEnable) — Thread-ID format: [0:0] - T0 or T1. N/A for GT/IO Cores [3:1] - Core-ID When 'TID filter enable' is clear; the specified counter will count All events Thread-ID of 0xF is reserved for non-associated requests like: - LLC victims - PMSeq - External snoops
18	package	0h	RW	Edge Detect Enable (EdgeDet) — Enable the counting on event edges (count occurrences) or level (number of clocks the event was high) 0 : level count 1 : Edge count
17	package	0h	WO	Counter Reset (CounterReset) — If we write 1 to this bit, it resets the appropriate PMON counter
16	package	0h	RW	Enable uBP (EnUBP) — Enable the generation of micro-breakpoint event on event occurrence. 0b Do not assert micro-breakpoint on event occurrence 1b Assert micro-breakpoint on event occurrence
15:8	package	0h	RW	Unit Mask (UnitMask) — A set of qualifiers defined for each of the events. Details for the counter are TBD.
7:0	package	0h	RW	Event Select (EvSlt) — a selector for measured signals Details of events are TBD

5.2.7 (706h) NcuPMONCtI2

Controls the operation of the the PMON Counte 2. All fields in this reigster must be reset when LT_CONTROL_MSR[LTPmonCtrClr] is set to 1b.

MSR Address: 706h				
Bit	Scope	Default	Attribute	Description
31:29	-	-	-	Reserved (RSVD) — Reserved.
28:24	package	0h	RW	Threshold — This field is compared directly against an incoming increment value from event increment bus. The width of this field is determined by the widest event in the IMPH (which is 5 bits for queue occupancy). The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the Invert bit) no matter how wide the original event was.
23	package	0h	RW	Intert Event (Invert) — Invert the event for counting purposes : 0 : Count when the event is asserted or on it's rising edge 1. : Count when the event is de-asserted or on it's falling edge
22	package	0h	RW	Counter Enable (CounterEn) — Enable the counter to count when the global enable is active. 0 : counter is disabled and will not count when global enable is set. 1 : counter is enabled and will count when global enable is set.
21	-	-	-	Reserved (RSVD) — Reserved.

continued...



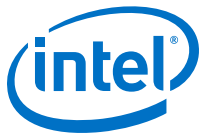
MSR Address: 706h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEn) — Enable generation of Overflow indication when this counter overflows. 0 : PMI Event for this counter is disabled 1 : PMI event is enabled
19	package	0h	RW	TID filter enable (TidFilterEnable) — Thread-ID format: [0:0] - T0 or T1. N/A for GT/IO Cores [3:1] - Core-ID When 'TID filter enable' is clear; the specified counter will count All events Thread-ID of 0xF is reserved for non-associated requests like: - LLC victims - PMSeq - External snoops
18	package	0h	RW	Edge Detect Enable (EdgeDet) — Enable the counting on event edges (count occurrences) or level (number of clocks the event was high) 0 : level count 1 : Edge count
17	package	0h	WO	Counter Reset (CounterReset) — If we write 1 to this bit, it resets the appropriate PMON counter
16	package	0h	RW	Enable uBP (EnUBP) — Enable the generation of micro-breakpoint event on event occurrence. 0b Do not assert micro-breakpoint on event occurrence 1b Assert micro-breakpoint on event occurrence
15:8	package	0h	RW	Unit Mask (UnitMask) — A set of qualifiers defined for each of the events. Details for the counter are TBD.
7:0	package	0h	RW	Event Select (EvSict) — a selector for measured signals Details of events are TBD

5.2.8 (708h) NcuPMONCtSts

NCU PMON Status register. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCntClr] is set to 1b.

MSR Address: 708h				
Bit	Scope	Default	Attribute	Description
31:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	RW1C	Overflow Counter 1 (OvfCnt1) — Indicates overflow Condition on the NCU Counter 1. Read 0 - counter didn't overflow 1 - counter overflow occurred. Write 0 - ignored
0	package	0h	RW1C	Overflow Counter 0 (OvfCnt0) — Indicates overflow Condition on the NCU Counter 0. Read

continued...



MSR Address: 708h				
Bit	Scope	Default	Attribute	Description
				0 - counter didn't overflow 1 - counter overflow occurred. Write 0 - ignored

5.2.9 (709h) NcuPMONCt1

PMON Counter 1 data register The Fix counter is counting UCLK cycles. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCntClr] is set to 1b.

MSR Address: 709h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	PMON Counter Data (PmonCtrData) — Pmon Counter data. The register may be read and written when the counter is stopped.

5.2.10 (70Ah) NcuPMONCt2

PMON Counter 2 data register The Fix counter is counting UCLK cycles. All fields in this register must be reset when LT_CONTROL_MSR[LTPmonCntClr] is set to 1b.

MSR Address: 70Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	PMON Counter Data (PmonCtrData) — Pmon Counter data. The register may be read and written when the counter is stopped.

5.2.11 (710h) PmonUnitCtrl

MSR Address: 710h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to
continued...				

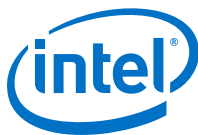


MSR Address: 710h				
Bit	Scope	Default	Attribute	Description
				freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	RW_V	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. This is a WO register, and writing to this bit will trigger a reset for 1 cycle only.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. This is a WO register, and writing to this bit will trigger a reset for 1 cycle only.

5.2.12 (711h) PmonCntrCfg_0

Perfmon Counter Control Register

MSR Address: 711h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW_V	Occupancy Edge Detect (OccEdgeDetect) — Enable edge detection on the output of the occupancy threshold comparison. When enabled and used with the Threshold logic, this will provide the ability to track the number of times when an occupancy counter exceed or was equal to the value in the threshold. When used with OccInvert, this will track the number of times when the value dropped below the threshold.
30	package	0h	RW_V	Occupancy Invert (OccInvert) — Invert the output of the occupancy threshold comparison logic. 0 -- "greater than or equal to" comparison on the threshold compare 1 -- "less than" comparison on the threshold compare
29:24	package	0h	RW_V	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison
continued...				



MSR Address: 711h				
Bit	Scope	Default	Attribute	Description
				depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW_V	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW_V	Counter Enable (CounterEnable) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW_V	Overflow Enable (OverflowEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW_V	Edge Detect (EdgeDetect) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
continued...				



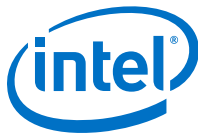
MSR Address: 711h				
Bit	Scope	Default	Attribute	Description
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	-	-	-	Reserved (RSVD) — Reserved.
15:14	package	0h	RW_V	Occupancy Select (OccSelect) — Select which of the four occupancy counters to use. 00 - Threads in C0 - KNL - won't be used in KNL 01 - Cores in C0 (Refer Tile in KNL) 10 - Cores in C3 - KNL - won't be used in KNL 11 - Cores in C6 (Refer Tile in KNL)
13:8	-	-	-	Reserved (RSVD) — Reserved.
7	package	0h	RW_V	Use the Occupancy Counter (UseOccupancy) — When enabled, this will pass the output of the occupancy block through to the counter (dependent on the event occurring).
6:0	package	0h	RW_V	Event Select (EventSelect) — This field is used to decode the PerfMon event which is selected.

5.2.13 (712h) PmonCntrCfg_1

Perfmon Counter Control Register

MSR Address: 712h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW_V	Occupancy Edge Detect (OccEdgeDetect) — Enable edge detection on the output of the occupancy threshold comparison. When enabled and used with the Threshold logic, this will provide the ability to track the number of times when an occupancy counter exceed or was equal to the value in the threshold. When used with OccInvert, this will track the number of times when the value dropped below the threshold.
30	package	0h	RW_V	Occupancy Invert (OccInvert) — Invert the output of the occupancy threshold comparison logic. 0 -- "greater than or equal to" comparison on the threshold compare 1 -- "less than" comparison on the threshold compare
29:24	package	0h	RW_V	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how

continued...



MSR Address: 712h				
Bit	Scope	Default	Attribute	Description
				wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW_V	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold $\neq 0$. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW_V	Counter Enable (CounterEnable) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW_V	Overflow Enable (OverflowEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW_V	Edge Detect (EdgeDetect) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
continued...				



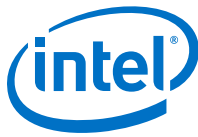
MSR Address: 712h				
Bit	Scope	Default	Attribute	Description
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	-	-	-	Reserved (RSVD) — Reserved.
15:14	package	0h	RW_V	Occupancy Select (OccSelect) — Select which of the four occupancy counters to use. 00 - Threads in C0 01 - Cores in C0 10 - Cores in C3 11 - Cores in C6
13:8	-	-	-	Reserved (RSVD) — Reserved.
7	package	0h	RW_V	Use the Occupancy Counter (UseOccupancy) — When enabled, this will pass the output of the occupancy block through to the counter (dependent on the event occurring).
6:0	package	0h	RW_V	Event Select (EventSelect) — This field is used to decode the PerfMon event which is selected.

5.2.14 (713h) PmonCntrCfg_2

Perfmon Counter Control Register

MSR Address: 713h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW_V	Occupancy Edge Detect (OccEdgeDetect) — Enable edge detection on the output of the occupancy threshold comparison. When enabled and used with the Threshold logic, this will provide the ability to track the number of times when an occupancy counter exceed or was equal to the value in the threshold. When used with OccInvert, this will track the number of times when the value dropped below the threshold.
30	package	0h	RW_V	Occupancy Invert (OccInvert) — Invert the output of the occupancy threshold comparison logic. 0 -- "greater than or equal to" comparison on the threshold compare 1 -- "less than" comparison on the threshold compare
29:24	package	0h	RW_V	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how

continued...



MSR Address: 713h				
Bit	Scope	Default	Attribute	Description
				wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW_V	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold $\neq 0$. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW_V	Counter Enable (CounterEnable) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW_V	Overflow Enable (OverflowEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW_V	Edge Detect (EdgeDetect) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
continued...				



MSR Address: 713h				
Bit	Scope	Default	Attribute	Description
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	-	-	-	Reserved (RSVD) — Reserved.
15:14	package	0h	RW_V	Occupancy Select (OccSelect) — Select which of the four occupancy counters to use. 00 - Threads in C0 01 - Cores in C0 10 - Cores in C3 11 - Cores in C6
13:8	-	-	-	Reserved (RSVD) — Reserved.
7	package	0h	RW_V	Use the Occupancy Counter (UseOccupancy) — When enabled, this will pass the output of the occupancy block through to the counter (dependent on the event occurring).
6:0	package	0h	RW_V	Event Select (EventSelect) — This field is used to decode the PerfMon event which is selected.

5.2.15 (714h) PmonCntrCfg_3

Perfmon Counter Control Register

MSR Address: 714h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW_V	Occupancy Edge Detect (OccEdgeDetect) — Enable edge detection on the output of the occupancy threshold comparison. When enabled and used with the Threshold logic, this will provide the ability to track the number of times when an occupancy counter exceed or was equal to the value in the threshold. When used with OccInvert, this will track the number of times when the value dropped below the threshold.
30	package	0h	RW_V	Occupancy Invert (OccInvert) — Invert the output of the occupancy threshold comparison logic. 0 -- "greater than or equal to" comparison on the threshold compare 1 -- "less than" comparison on the threshold compare
29:24	package	0h	RW_V	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how

continued...



MSR Address: 714h				
Bit	Scope	Default	Attribute	Description
				wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW_V	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW_V	Counter Enable (CounterEnable) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW_V	Overflow Enable (OverflowEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW_V	Edge Detect (EdgeDetect) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
continued...				



MSR Address: 714h				
Bit	Scope	Default	Attribute	Description
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	-	-	-	Reserved (RSVD) — Reserved.
15:14	package	0h	RW_V	Occupancy Select (OccSelect) — Select which of the four occupancy counters to use. 00 - Threads in C0 01 - Cores in C0 10 - Cores in C3 11 - Cores in C6
13:8	-	-	-	Reserved (RSVD) — Reserved.
7	package	0h	RW_V	Use the Occupancy Counter (UseOccupancy) — When enabled, this will pass the output of the occupancy block through to the counter (dependent on the event occurring).
6:0	package	0h	RW_V	Event Select (EventSelect) — This field is used to decode the PerfMon event which is selected.

5.2.16 (717h) PmonCntr_0

This register is a perfmon counter. Software can both read it and write it.

MSR Address: 717h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (CounterValue) — This is the current value of the counter.

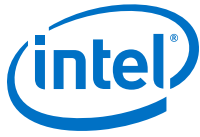
5.2.17 (718h) PmonCntr_1

This register is a perfmon counter. Software can both read it and write it.

MSR Address: 718h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (CounterValue) — This is the current value of the counter.

5.2.18 (719h) PmonCntr_2

This register is a perfmon counter. Software can both read it and write it.



MSR Address: 719h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (CounterValue) — This is the current value of the counter.

5.2.19 (71Ah) PmonCntn_3

This register is a perfmon counter. Software can both read it and write it.

MSR Address: 71Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (CounterValue) — This is the current value of the counter.

5.2.20 (E00h) PERF_CTR_UNIT_CTRL_CHA_0

Pmon Unit control.

MSR Address: E00h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.



5.2.21 (E01h) PERF_EVT_SEL_0_CHA_0

Perfmon Counter Control Register

MSR Address: E01h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: E01h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.22 (E02h) PERF_EVT_SEL_1_CHA_0

Perfmon Counter Control Register

MSR Address: E02h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

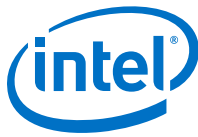


MSR Address: E02h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.23 (E03h) PERF_EVT_SEL_2_CHA_0

Perfmon Counter Control Register

MSR Address: E03h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: E03h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.24 (E04h) PERF_EVT_SEL_3_CHA_0

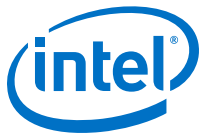
Perfmon Counter Control Register

MSR Address: E04h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see

continued...



MSR Address: E04h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				



MSR Address: E04h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.25 (E05h) PERF_UNIT_CTL_CHA_0

MSR Address: E05h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.26 (E06h) PERF_UNIT_CTL_1_CHA_0

MSR Address: E06h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
continued...				



MSR Address: E06h				
Bit	Scope	Default	Attribute	Description
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.27 (E07h) PERF_UNIT_STATUS_CHA_0

Cbo Pmon counters status

MSR Address: E07h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

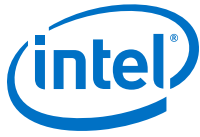
5.2.28 (E08h) PERF_CTR_0_CHA_0

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E08h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.29 (E09h) PERF_CTR_1_CHA_0

This register is a perfmon counter. Software can both read it and write it.



MSR Address: E09h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.30 (E0Ah) PERF_CTR_2_CHA_0

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E0Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.31 (E0Bh) PERF_CTR_3_CHA_0

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E0Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.32 (E0Ch) PERF_CTR_UNIT_CTRL_CHA_1

Pmon Unit control.

MSR Address: E0Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
continued...				

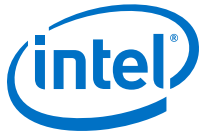


MSR Address: E0Ch				
Bit	Scope	Default	Attribute	Description
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.33 (E0Dh) PERF_EVT_SEL_0_CHA_1

Perfmon Counter Control Register

MSR Address: E0Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge
continued...				



MSR Address: E0Dh				
Bit	Scope	Default	Attribute	Description
				detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.34 (E0Eh) PERF_EVT_SEL_1_CHA_1

Perfmon Counter Control Register

MSR Address: E0Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q,
continued...				

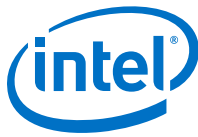


MSR Address: E0Eh				
Bit	Scope	Default	Attribute	Description
				\qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.35 (E0Fh) PERF_EVT_SEL_2_CHA_1

Perfmon Counter Control Register

MSR Address: E0Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is
continued...				



MSR Address: E0Fh				
Bit	Scope	Default	Attribute	Description
				unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				



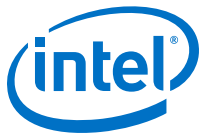
MSR Address: E0Fh				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.36 (E10h) PERF_EVT_SEL_3_CHA_1

Perfmon Counter Control Register

MSR Address: E10h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.

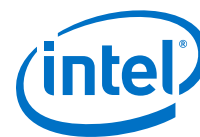
continued...



MSR Address: E10h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.37 (E11h) PERF_UNIT_CTL_CHA_1

MSR Address: E11h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID



5.2.38 (E12h) PERF_UNIT_CTL_1_CHA_1

MSR Address: E12h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

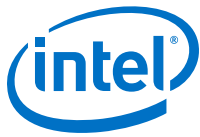
5.2.39 (E13h) PERF_UNIT_STATUS_CHA_1

Cbo Pmon counters status

MSR Address: E13h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.40 (E14h) PERF_CTR_0_CHA_1

This register is a perfmon counter. Software can both read it and write it.



MSR Address: E14h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.41 (E15h) PERF_CTR_1_CHA_1

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E15h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.42 (E16h) PERF_CTR_2_CHA_1

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E16h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.43 (E17h) PERF_CTR_3_CHA_1

This register is a perfmon counter. Software can both read it and write it.

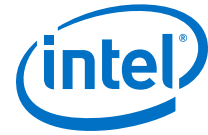
MSR Address: E17h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.44 (E18h) PERF_CTR_UNIT_CTRL_CHA_2

Pmon Unit control.

MSR Address: E18h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially

continued...

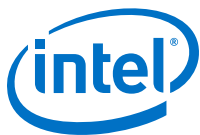


MSR Address: E18h				
Bit	Scope	Default	Attribute	Description
				triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.45 (E19h) PERF_EVT_SEL_0_CHA_2

Perfmon Counter Control Register

MSR Address: E19h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E19h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

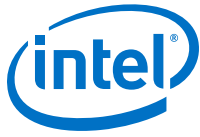
5.2.46 (E1Ah) PERF_EVT_SEL_1_CHA_2

Perfmon Counter Control Register

MSR Address: E1Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E1Ah				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.47 (E1Bh) PERF_EVT_SEL_2_CHA_2

Perfmon Counter Control Register

MSR Address: E1Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				



MSR Address: E1Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.48 (E1Ch) PERF_EVT_SEL_3_CHA_2

Perfmon Counter Control Register

MSR Address: E1Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: E1Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.49 (E1Dh) PERF_UNIT_CTL_CHA_2

MSR Address: E1Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E1Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.50 (E1Eh) PERF_UNIT_CTL_1_CHA_2

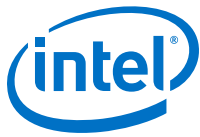
MSR Address: E1Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.51 (E1Fh) PERF_UNIT_STATUS_CHA_2

Cbo Pmon counters status

MSR Address: E1Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed

continued...



MSR Address: E1Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.52 (E20h) PERF_CTR_0_CHA_2

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E20h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.53 (E21h) PERF_CTR_1_CHA_2

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E21h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.54 (E22h) PERF_CTR_2_CHA_2

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E22h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.55 (E23h) PERF_CTR_3_CHA_2

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E23h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.56 (E24h) PERF_CTR_UNIT_CTRL_CHA_3

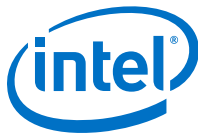
Pmon Unit control.

MSR Address: E24h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.57 (E25h) PERF_EVT_SEL_0_CHA_3

Perfmon Counter Control Register

MSR Address: E25h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: E25h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

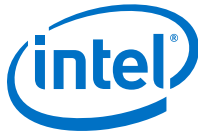
5.2.58 (E26h) PERF_EVT_SEL_1_CHA_3

Perfmon Counter Control Register



MSR Address: E26h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: E26h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.59 (E27h) PERF_EVT_SEL_2_CHA_3

Perfmon Counter Control Register

MSR Address: E27h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

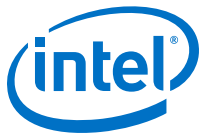


MSR Address: E27h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.60 (E28h) PERF_EVT_SEL_3_CHA_3

Perfmon Counter Control Register

MSR Address: E28h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: E28h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.61 (E29h) PERF_UNIT_CTL_CHA_3

MSR Address: E29h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved

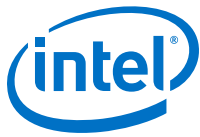
continued...



MSR Address: E29h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.62 (E2Ah) PERF_UNIT_CTL_1_CHA_3

MSR Address: E2Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.63 (E2Bh) PERF_UNIT_STATUS_CHA_3

Cbo Pmon counters status

MSR Address: E2Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.64 (E2Ch) PERF_CTR_0_CHA_3

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E2Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.65 (E2Dh) PERF_CTR_1_CHA_3

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E2Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

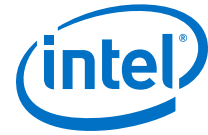
5.2.66 (E2Eh) PERF_CTR_2_CHA_3

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E2Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.67 (E2Fh) PERF_CTR_3_CHA_3

This register is a perfmon counter. Software can both read it and write it.



MSR Address: E2Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.68 (E30h) PERF_CTR_UNIT_CTRL_CHA_4

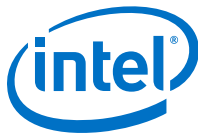
Pmon Unit control.

MSR Address: E30h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.69 (E31h) PERF_EVT_SEL_0_CHA_4

Perfmon Counter Control Register

MSR Address: E31h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: E31h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected

continued...



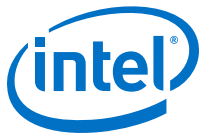
MSR Address: E31h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.70 (E32h) PERF_EVT_SEL_1_CHA_4

Perfmon Counter Control Register

MSR Address: E32h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.

continued...



MSR Address: E32h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.71 (E33h) PERF_EVT_SEL_2_CHA_4

Perfmon Counter Control Register

MSR Address: E33h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E33h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.72 (E34h) PERF_EVT_SEL_3_CHA_4

Perfmon Counter Control Register

MSR Address: E34h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E34h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

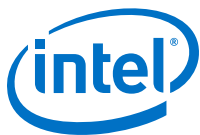


5.2.73 (E35h) PERF_UNIT_CTL_CHA_4

MSR Address: E35h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.74 (E36h) PERF_UNIT_CTL_1_CHA_4

MSR Address: E36h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: E36h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.75 (E37h) PERF_UNIT_STATUS_CHA_4

Cbo Pmon counters status

MSR Address: E37h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.76 (E38h) PERF_CTR_0_CHA_4

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E38h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.77 (E39h) PERF_CTR_1_CHA_4

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E39h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.78 (E3Ah) PERF_CTR_2_CHA_4

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E3Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.79 (E3Bh) PERF_CTR_3_CHA_4

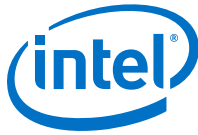
This register is a perfmon counter. Software can both read it and write it.

MSR Address: E3Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.80 (E3Ch) PERF_CTR_UNIT_CTRL_CHA_5

Pmon Unit control.

MSR Address: E3Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				



MSR Address: E3Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.81 (E3Dh) PERF_EVT_SEL_0_CHA_5

Perfmon Counter Control Register

MSR Address: E3Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				

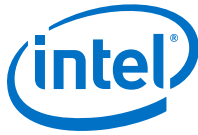


MSR Address: E3Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.82 (E3Eh) PERF_EVT_SEL_1_CHA_5

Perfmon Counter Control Register

MSR Address: E3Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E3Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

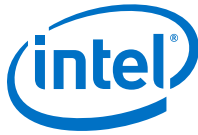
5.2.83 (E3Fh) PERF_EVT_SEL_2_CHA_5

Perfmon Counter Control Register

MSR Address: E3Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E3Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.84 (E40h) PERF_EVT_SEL_3_CHA_5

Perfmon Counter Control Register

MSR Address: E40h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				

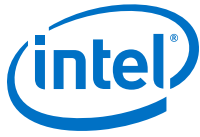


MSR Address: E40h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.85 (E41h) PERF_UNIT_CTL_CHA_5

MSR Address: E41h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.86 (E42h) PERF_UNIT_CTL_1_CHA_5



MSR Address: E42h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.87 (E43h) PERF_UNIT_STATUS_CHA_5

Cbo Pmon counters status

MSR Address: E43h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.88 (E44h) PERF_CTR_0_CHA_5

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E44h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.89 (E45h) PERF_CTR_1_CHA_5

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E45h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.90 (E46h) PERF_CTR_2_CHA_5

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E46h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.91 (E47h) PERF_CTR_3_CHA_5

This register is a perfmon counter. Software can both read it and write it.

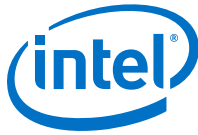
MSR Address: E47h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.92 (E48h) PERF_CTR_UNIT_CTRL_CHA_6

Pmon Unit control.

MSR Address: E48h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.

continued...



MSR Address: E48h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.93 (E49h) PERF_EVT_SEL_0_CHA_6

Perfmon Counter Control Register

MSR Address: E49h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				

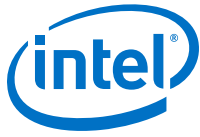


MSR Address: E49h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.94 (E4Ah) PERF_EVT_SEL_1_CHA_6

Perfmon Counter Control Register

MSR Address: E4Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: E4Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

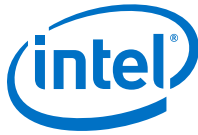
5.2.95 (E4Bh) PERF_EVT_SEL_2_CHA_6

Perfmon Counter Control Register



MSR Address: E4Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: E4Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.96 (E4Ch) PERF_EVT_SEL_3_CHA_6

Perfmon Counter Control Register

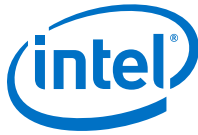
MSR Address: E4Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: E4Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.97 (E4Dh) PERF_UNIT_CTL_CHA_6

MSR Address: E4Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E4Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format: [2:0] - ThreadId. [8:3] - Core-ID

5.2.98 (E4Eh) PERF_UNIT_CTL_1_CHA_6

MSR Address: E4Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.99 (E4Fh) PERF_UNIT_STATUS_CHA_6

Cbo Pmon counters status

MSR Address: E4Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
continued...				



MSR Address: E4Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.100 (E50h) PERF_CTR_0_CHA_6

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E50h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.101 (E51h) PERF_CTR_1_CHA_6

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E51h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.102 (E52h) PERF_CTR_2_CHA_6

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E52h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.103 (E53h) PERF_CTR_3_CHA_6

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E53h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.104 (E54h) PERF_CTR_UNIT_CTRL_CHA_7

Pmon Unit control.

MSR Address: E54h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.105 (E55h) PERF_EVT_SEL_0_CHA_7

Perfmon Counter Control Register

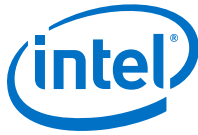
MSR Address: E55h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: E55h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.106 (E56h) PERF_EVT_SEL_1_CHA_7

Perfmon Counter Control Register



MSR Address: E56h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

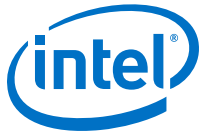


MSR Address: E56h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.107 (E57h) PERF_EVT_SEL_2_CHA_7

Perfmon Counter Control Register

MSR Address: E57h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: E57h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.108 (E58h) PERF_EVT_SEL_3_CHA_7

Perfmon Counter Control Register

MSR Address: E58h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				

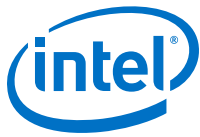


MSR Address: E58h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.109 (E59h) PERF_UNIT_CTL_CHA_7

MSR Address: E59h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved

continued...



MSR Address: E59h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.110 (E5Ah) PERF_UNIT_CTL_1_CHA_7

MSR Address: E5Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.111 (E5Bh) PERF_UNIT_STATUS_CHA_7

Cbo Pmon counters status

MSR Address: E5Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.112 (E5Ch) PERF_CTR_0_CHA_7

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E5Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.113 (E5Dh) PERF_CTR_1_CHA_7

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E5Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

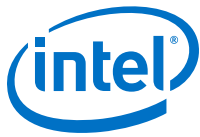
5.2.114 (E5Eh) PERF_CTR_2_CHA_7

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E5Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.115 (E5Fh) PERF_CTR_3_CHA_7

This register is a perfmon counter. Software can both read it and write it.



MSR Address: E5Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.116 (E60h) PERF_CTR_UNIT_CTRL_CHA_8

Pmon Unit control.

MSR Address: E60h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.117 (E61h) PERF_EVT_SEL_0_CHA_8

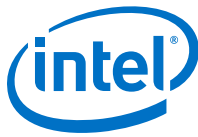
Perfmon Counter Control Register

MSR Address: E61h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E61h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected

continued...



MSR Address: E61h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.118 (E62h) PERF_EVT_SEL_1_CHA_8

Perfmon Counter Control Register

MSR Address: E62h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.
continued...				

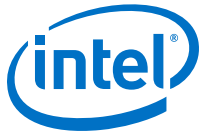


MSR Address: E62h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.119 (E63h) PERF_EVT_SEL_2_CHA_8

Perfmon Counter Control Register

MSR Address: E63h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E63h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.120 (E64h) PERF_EVT_SEL_3_CHA_8

Perfmon Counter Control Register

MSR Address: E64h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E64h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.121 (E65h) PERF_UNIT_CTL_CHA_8

MSR Address: E65h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.122 (E66h) PERF_UNIT_CTL_1_CHA_8

MSR Address: E66h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: E66h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.123 (E67h) PERF_UNIT_STATUS_CHA_8

Cbo Pmon counters status

MSR Address: E67h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.124 (E68h) PERF_CTR_0_CHA_8

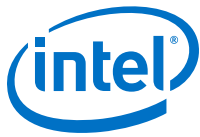
This register is a perfmon counter. Software can both read it and write it.

MSR Address: E68h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.125 (E69h) PERF_CTR_1_CHA_8

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E69h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.126 (E6Ah) PERF_CTR_2_CHA_8

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E6Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.127 (E6Bh) PERF_CTR_3_CHA_8

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E6Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.128 (E6Ch) PERF_CTR_UNIT_CTRL_CHA_9

Pmon Unit control.

MSR Address: E6Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				

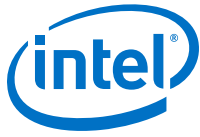


MSR Address: E6Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.129 (E6Dh) PERF_EVT_SEL_0_CHA_9

Perfmon Counter Control Register

MSR Address: E6Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				



MSR Address: E6Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.130 (E6Eh) PERF_EVT_SEL_1_CHA_9

Perfmon Counter Control Register

MSR Address: E6Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E6Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.131 (E6Fh) PERF_EVT_SEL_2_CHA_9

Perfmon Counter Control Register

MSR Address: E6Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E6Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold \leq event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\event select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

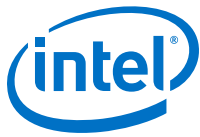


5.2.132 (E70h) PERF_EVT_SEL_3_CHA_9

Perfmon Counter Control Register

MSR Address: E70h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: E70h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.133 (E71h) PERF_UNIT_CTL_CHA_9

MSR Address: E71h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.134 (E72h) PERF_UNIT_CTL_1_CHA_9



MSR Address: E72h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.135 (E73h) PERF_UNIT_STATUS_CHA_9

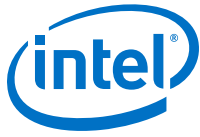
Cbo Pmon counters status

MSR Address: E73h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.136 (E74h) PERF_CTR_0_CHA_9

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E74h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.137 (E75h) PERF_CTR_1_CHA_9

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E75h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.138 (E76h) PERF_CTR_2_CHA_9

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E76h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.139 (E77h) PERF_CTR_3_CHA_9

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E77h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.140 (E78h) PERF_CTR_UNIT_CTRL_CHA_10

Pmon Unit control.

MSR Address: E78h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
continued...				

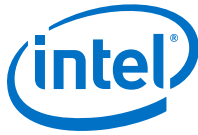


MSR Address: E78h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.141 (E79h) PERF_EVT_SEL_0_CHA_10

Perfmon Counter Control Register

MSR Address: E79h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				



MSR Address: E79h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.142 (E7Ah) PERF_EVT_SEL_1_CHA_10

Perfmon Counter Control Register

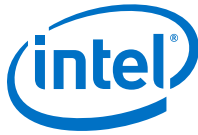
MSR Address: E7Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: E7Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.143 (E7Bh) PERF_EVT_SEL_2_CHA_10

Perfmon Counter Control Register



MSR Address: E7Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

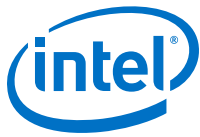


MSR Address: E7Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.144 (E7Ch) PERF_EVT_SEL_3_CHA_10

Perfmon Counter Control Register

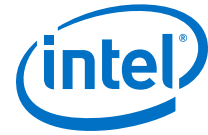
MSR Address: E7Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: E7Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.145 (E7Dh) PERF_UNIT_CTL_CHA_10

MSR Address: E7Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E7Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.146 (E7Eh) PERF_UNIT_CTL_1_CHA_10

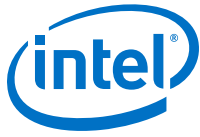
MSR Address: E7Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.147 (E7Fh) PERF_UNIT_STATUS_CHA_10

Cbo Pmon counters status

MSR Address: E7Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed

continued...



MSR Address: E7Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.148 (E80h) PERF_CTR_0_CHA_10

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E80h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.149 (E81h) PERF_CTR_1_CHA_10

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E81h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.150 (E82h) PERF_CTR_2_CHA_10

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E82h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.151 (E83h) PERF_CTR_3_CHA_10

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E83h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.152 (E84h) PERF_CTR_UNIT_CTRL_CHA_11

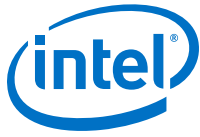
Pmon Unit control.

MSR Address: E84h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.153 (E85h) PERF_EVT_SEL_0_CHA_11

Perfmon Counter Control Register

MSR Address: E85h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: E85h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

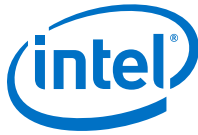
5.2.154 (E86h) PERF_EVT_SEL_1_CHA_11

Perfmon Counter Control Register



MSR Address: E86h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: E86h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.155 (E87h) PERF_EVT_SEL_2_CHA_11

Perfmon Counter Control Register

MSR Address: E87h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

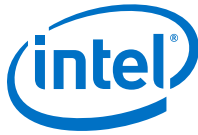


MSR Address: E87h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.156 (E88h) PERF_EVT_SEL_3_CHA_11

Perfmon Counter Control Register

MSR Address: E88h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: E88h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.157 (E89h) PERF_UNIT_CTL_CHA_11

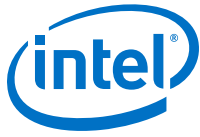
MSR Address: E89h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved
continued...				



MSR Address: E89h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.158 (E8Ah) PERF_UNIT_CTL_1_CHA_11

MSR Address: E8Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.159 (E8Bh) PERF_UNIT_STATUS_CHA_11

Cbo Pmon counters status

MSR Address: E8Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.160 (E8Ch) PERF_CTR_0_CHA_11

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E8Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.161 (E8Dh) PERF_CTR_1_CHA_11

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E8Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.162 (E8Eh) PERF_CTR_2_CHA_11

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E8Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.163 (E8Fh) PERF_CTR_3_CHA_11

This register is a perfmon counter. Software can both read it and write it.



MSR Address: E8Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.164 (E90h) PERF_CTR_UNIT_CTRL_CHA_12

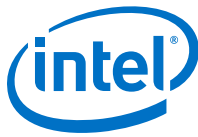
Pmon Unit control.

MSR Address: E90h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.165 (E91h) PERF_EVT_SEL_0_CHA_12

Perfmon Counter Control Register

MSR Address: E91h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: E91h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				

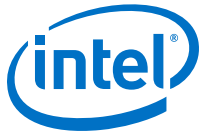


MSR Address: E91h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.166 (E92h) PERF_EVT_SEL_1_CHA_12

Perfmon Counter Control Register

MSR Address: E92h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.
continued...				



MSR Address: E92h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.167 (E93h) PERF_EVT_SEL_2_CHA_12

Perfmon Counter Control Register

MSR Address: E93h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				

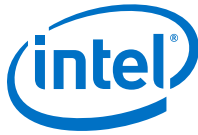


MSR Address: E93h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.168 (E94h) PERF_EVT_SEL_3_CHA_12

Perfmon Counter Control Register

MSR Address: E94h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E94h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold \leq event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\event select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

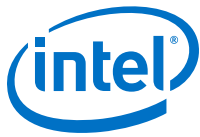


5.2.169 (E95h) PERF_UNIT_CTL_CHA_12

MSR Address: E95h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.170 (E96h) PERF_UNIT_CTL_1_CHA_12

MSR Address: E96h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: E96h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.171 (E97h) PERF_UNIT_STATUS_CHA_12

Cbo Pmon counters status

MSR Address: E97h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.172 (E98h) PERF_CTR_0_CHA_12

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E98h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.173 (E99h) PERF_CTR_1_CHA_12

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E99h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.174 (E9Ah) PERF_CTR_2_CHA_12

This register is a perfmon counter. Software can both read it and write it.

MSR Address: E9Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.175 (E9Bh) PERF_CTR_3_CHA_12

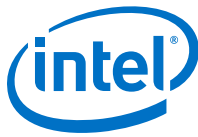
This register is a perfmon counter. Software can both read it and write it.

MSR Address: E9Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.176 (E9Ch) PERF_CTR_UNIT_CTRL_CHA_13

Pmon Unit control.

MSR Address: E9Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				



MSR Address: E9Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.177 (E9Dh) PERF_EVT_SEL_0_CHA_13

Perfmon Counter Control Register

MSR Address: E9Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				



MSR Address: E9Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.178 (E9Eh) PERF_EVT_SEL_1_CHA_13

Perfmon Counter Control Register

MSR Address: E9Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: E9Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.179 (E9Fh) PERF_EVT_SEL_2_CHA_13

Perfmon Counter Control Register

MSR Address: E9Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: E9Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.180 (EA0h) PERF_EVT_SEL_3_CHA_13

Perfmon Counter Control Register

MSR Address: EA0h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				

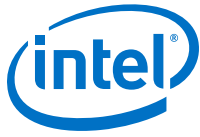


MSR Address: EA0h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.181 (EA1h) PERF_UNIT_CTL_CHA_13

MSR Address: EA1h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.182 (EA2h) PERF_UNIT_CTL_1_CHA_13



MSR Address: EA2h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.183 (EA3h) PERF_UNIT_STATUS_CHA_13

Cbo Pmon counters status

MSR Address: EA3h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.184 (EA4h) PERF_CTR_0_CHA_13

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EA4h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.185 (EA5h) PERF_CTR_1_CHA_13

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EA5h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.186 (EA6h) PERF_CTR_2_CHA_13

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EA6h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.187 (EA7h) PERF_CTR_3_CHA_13

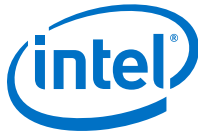
This register is a perfmon counter. Software can both read it and write it.

MSR Address: EA7h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.188 (EA8h) PERF_CTR_UNIT_CTRL_CHA_14

Pmon Unit control.

MSR Address: EA8h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
<i>continued...</i>				



MSR Address: EA8h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.189 (EA9h) PERF_EVT_SEL_0_CHA_14

Perfmon Counter Control Register

MSR Address: EA9h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				

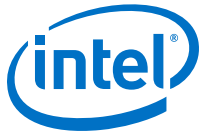


MSR Address: EA9h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.190 (EAAh) PERF_EVT_SEL_1_CHA_14

Perfmon Counter Control Register

MSR Address: EAAh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: EAAh				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

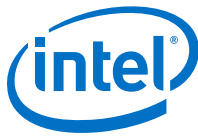
5.2.191 (EABh) PERF_EVT_SEL_2_CHA_14

Perfmon Counter Control Register



MSR Address: EABh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: EABh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.192 (EAcH) PERF_EVT_SEL_3_CHA_14

Perfmon Counter Control Register

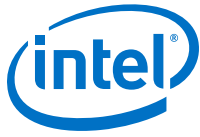
MSR Address: EAcH				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: EACb				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.193 (EADb) PERF_UNIT_CTL_CHA_14

MSR Address: EADb				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: EADh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.194 (EAEh) PERF_UNIT_CTL_1_CHA_14

MSR Address: EAEh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.195 (EAFh) PERF_UNIT_STATUS_CHA_14

Cbo Pmon counters status

MSR Address: EAFh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
continued...				



MSR Address: EAFh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.196 (EB0h) PERF_CTR_0_CHA_14

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EB0h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.197 (EB1h) PERF_CTR_1_CHA_14

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EB1h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.198 (EB2h) PERF_CTR_2_CHA_14

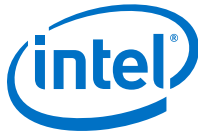
This register is a perfmon counter. Software can both read it and write it.

MSR Address: EB2h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.199 (EB3h) PERF_CTR_3_CHA_14

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EB3h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.200 (EB4h) PERF_CTR_UNIT_CTRL_CHA_15

Pmon Unit control.

MSR Address: EB4h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.201 (EB5h) PERF_EVT_SEL_0_CHA_15

Perfmon Counter Control Register

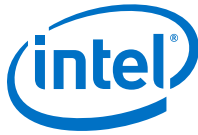
MSR Address: EB5h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: EB5h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.202 (EB6h) PERF_EVT_SEL_1_CHA_15

Perfmon Counter Control Register



MSR Address: EB6h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

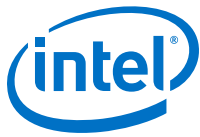


MSR Address: EB6h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.203 (EB7h) PERF_EVT_SEL_2_CHA_15

Perfmon Counter Control Register

MSR Address: EB7h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: EB7h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.204 (EB8h) PERF_EVT_SEL_3_CHA_15

Perfmon Counter Control Register

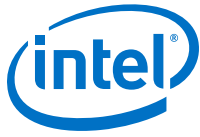
MSR Address: EB8h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: EB8h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.205 (EB9h) PERF_UNIT_CTL_CHA_15

MSR Address: EB9h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved
continued...				



MSR Address: EB9h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.206 (EBAh) PERF_UNIT_CTL_1_CHA_15

MSR Address: EBAh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.207 (EBBh) PERF_UNIT_STATUS_CHA_15

Cbo Pmon counters status

MSR Address: EBBh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.208 (EBCh) PERF_CTR_0_CHA_15

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EBCh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.209 (EBDh) PERF_CTR_1_CHA_15

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EBDh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

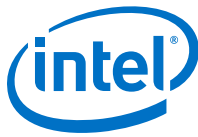
5.2.210 (EBEh) PERF_CTR_2_CHA_15

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EBEh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.211 (EBFh) PERF_CTR_3_CHA_15

This register is a perfmon counter. Software can both read it and write it.



MSR Address: EBFh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.212 (EC0h) PERF_CTR_UNIT_CTRL_CHA_16

Pmon Unit control.

MSR Address: EC0h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.213 (EC1h) PERF_EVT_SEL_0_CHA_16

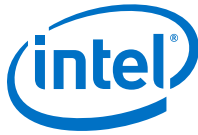
Perfmon Counter Control Register

MSR Address: EC1h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: EC1h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected

continued...



MSR Address: EC1h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.214 (EC2h) PERF_EVT_SEL_1_CHA_16

Perfmon Counter Control Register

MSR Address: EC2h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.
continued...				

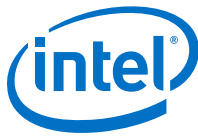


MSR Address: EC2h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.215 (EC3h) PERF_EVT_SEL_2_CHA_16

Perfmon Counter Control Register

MSR Address: EC3h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: EC3h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

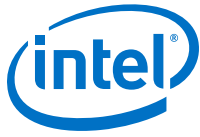
5.2.216 (EC4h) PERF_EVT_SEL_3_CHA_16

Perfmon Counter Control Register

MSR Address: EC4h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: EC4h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.217 (EC5h) PERF_UNIT_CTL_CHA_16

MSR Address: EC5h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.218 (EC6h) PERF_UNIT_CTL_1_CHA_16

MSR Address: EC6h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: EC6h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.219 (EC7h) PERF_UNIT_STATUS_CHA_16

Cbo Pmon counters status

MSR Address: EC7h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.220 (EC8h) PERF_CTR_0_CHA_16

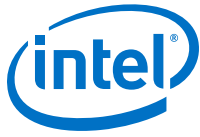
This register is a perfmon counter. Software can both read it and write it.

MSR Address: EC8h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.221 (EC9h) PERF_CTR_1_CHA_16

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EC9h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.222 (ECAh) PERF_CTR_2_CHA_16

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ECAh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.223 (ECBh) PERF_CTR_3_CHA_16

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ECBh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.224 (ECCh) PERF_CTR_UNIT_CTRL_CHA_17

Pmon Unit control.

MSR Address: ECCh				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				

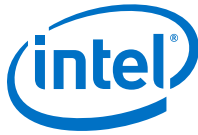


MSR Address: ECCh				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.225 (ECDh) PERF_EVT_SEL_0_CHA_17

Perfmon Counter Control Register

MSR Address: ECDh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				

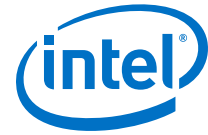


MSR Address: ECDh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.226 (ECEh) PERF_EVT_SEL_1_CHA_17

Perfmon Counter Control Register

MSR Address: ECEh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				

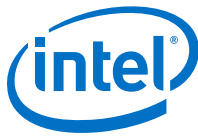


MSR Address: ECEh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.227 (ECFh) PERF_EVT_SEL_2_CHA_17

Perfmon Counter Control Register

MSR Address: ECFh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: ECFh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

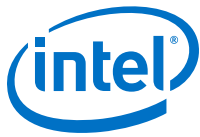


5.2.228 (ED0h) PERF_EVT_SEL_3_CHA_17

Perfmon Counter Control Register

MSR Address: ED0h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: ED0h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.229 (ED1h) PERF_UNIT_CTL_CHA_17

MSR Address: ED1h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.230 (ED2h) PERF_UNIT_CTL_1_CHA_17



MSR Address: ED2h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.231 (ED3h) PERF_UNIT_STATUS_CHA_17

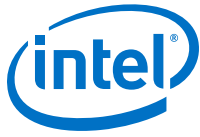
Cbo Pmon counters status

MSR Address: ED3h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.232 (ED4h) PERF_CTR_0_CHA_17

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ED4h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.233 (ED5h) PERF_CTR_1_CHA_17

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ED5h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.234 (ED6h) PERF_CTR_2_CHA_17

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ED6h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.235 (ED7h) PERF_CTR_3_CHA_17

This register is a perfmon counter. Software can both read it and write it.

MSR Address: ED7h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.236 (ED8h) PERF_CTR_UNIT_CTRL_CHA_18

Pmon Unit control.

MSR Address: ED8h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
continued...				

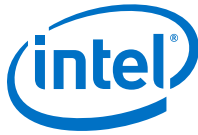


MSR Address: ED8h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.237 (ED9h) PERF_EVT_SEL_0_CHA_18

Perfmon Counter Control Register

MSR Address: ED9h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				



MSR Address: ED9h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.238 (EDAh) PERF_EVT_SEL_1_CHA_18

Perfmon Counter Control Register

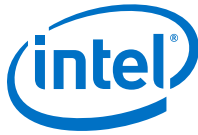
MSR Address: EDAh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: EDAh				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.239 (EDBh) PERF_EVT_SEL_2_CHA_18

Perfmon Counter Control Register



MSR Address: EDBh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

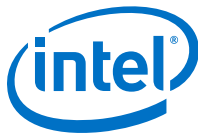


MSR Address: EDBh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.240 (EDCh) PERF_EVT_SEL_3_CHA_18

Perfmon Counter Control Register

MSR Address: EDCh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: EDCh				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.241 (EDDh) PERF_UNIT_CTL_CHA_18

MSR Address: EDDh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: EDDh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.242 (EDEh) PERF_UNIT_CTL_1_CHA_18

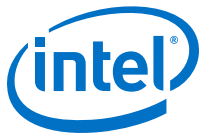
MSR Address: EDEh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.243 (EDFh) PERF_UNIT_STATUS_CHA_18

Cbo Pmon counters status

MSR Address: EDFh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed

continued...



MSR Address: EDFh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.244 (EE0h) PERF_CTR_0_CHA_18

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EE0h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.245 (EE1h) PERF_CTR_1_CHA_18

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EE1h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.246 (EE2h) PERF_CTR_2_CHA_18

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EE2h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.247 (EE3h) PERF_CTR_3_CHA_18

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EE3h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.248 (EE4h) PERF_CTR_UNIT_CTRL_CHA_19

Pmon Unit control.

MSR Address: EE4h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.249 (EE5h) PERF_EVT_SEL_0_CHA_19

Perfmon Counter Control Register

MSR Address: EE5h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: EE5h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.250 (EE6h) PERF_EVT_SEL_1_CHA_19

Perfmon Counter Control Register



MSR Address: EE6h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: EE6h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.251 (EE7h) PERF_EVT_SEL_2_CHA_19

Perfmon Counter Control Register

MSR Address: EE7h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

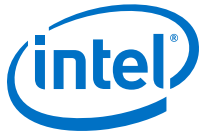


MSR Address: EE7h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.252 (EE8h) PERF_EVT_SEL_3_CHA_19

Perfmon Counter Control Register

MSR Address: EE8h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: EE8h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.253 (EE9h) PERF_UNIT_CTL_CHA_19

MSR Address: EE9h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved

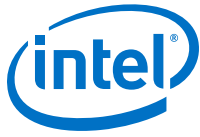
continued...



MSR Address: EE9h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.254 (EEAh) PERF_UNIT_CTL_1_CHA_19

MSR Address: EEAh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.255 (EEBh) PERF_UNIT_STATUS_CHA_19

Cbo Pmon counters status

MSR Address: EEBh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.256 (EECh) PERF_CTR_0_CHA_19

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EECh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.257 (EEDh) PERF_CTR_1_CHA_19

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EEDh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.258 (EEHh) PERF_CTR_2_CHA_19

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EEHh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.259 (EEFh) PERF_CTR_3_CHA_19

This register is a perfmon counter. Software can both read it and write it.



MSR Address: EEHh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.260 (EF0h) PERF_CTR_UNIT_CTRL_CHA_20

Pmon Unit control.

MSR Address: EF0h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.261 (EF1h) PERF_EVT_SEL_0_CHA_20

Perfmon Counter Control Register

MSR Address: EF1h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: EF1h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				



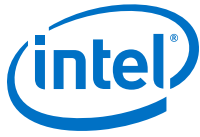
MSR Address: EF1h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.262 (EF2h) PERF_EVT_SEL_1_CHA_20

Perfmon Counter Control Register

MSR Address: EF2h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.

continued...



MSR Address: EF2h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.263 (EF3h) PERF_EVT_SEL_2_CHA_20

Perfmon Counter Control Register

MSR Address: EF3h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



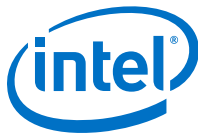
MSR Address: EF3h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

5.2.264 (EF4h) PERF_EVT_SEL_3_CHA_20

Perfmon Counter Control Register

MSR Address: EF4h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see

continued...



MSR Address: EF4h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

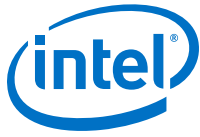


5.2.265 (EF5h) PERF_UNIT_CTL_CHA_20

MSR Address: EF5h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.266 (EF6h) PERF_UNIT_CTL_1_CHA_20

MSR Address: EF6h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: EF6h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.267 (EF7h) PERF_UNIT_STATUS_CHA_20

Cbo Pmon counters status

MSR Address: EF7h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.268 (EF8h) PERF_CTR_0_CHA_20

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EF8h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.269 (EF9h) PERF_CTR_1_CHA_20

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EF9h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.270 (EFAh) PERF_CTR_2_CHA_20

This register is a perfmon counter. Software can both read it and write it.

MSR Address: EFAh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.271 (EFBh) PERF_CTR_3_CHA_20

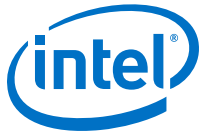
This register is a perfmon counter. Software can both read it and write it.

MSR Address: EFBh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.272 (EFCh) PERF_CTR_UNIT_CTRL_CHA_21

Pmon Unit control.

MSR Address: EFCh				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				



MSR Address: EFDh				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.273 (EFDh) PERF_EVT_SEL_0_CHA_21

Perfmon Counter Control Register

MSR Address: EFDh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				

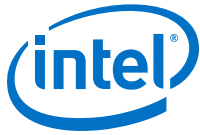


MSR Address: EFDh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.274 (EFEh) PERF_EVT_SEL_1_CHA_21

Perfmon Counter Control Register

MSR Address: EFEh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: EFHh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

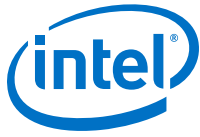
5.2.275 (EFFh) PERF_EVT_SEL_2_CHA_21

Perfmon Counter Control Register

MSR Address: EFFh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: EFFh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

**5.2.276 (F00h) PERF_EVT_SEL_3_CHA_21**

Perfmon Counter Control Register

MSR Address: F00h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				

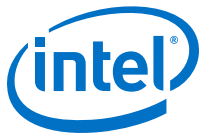


MSR Address: F00h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.277 (F01h) PERF_UNIT_CTL_CHA_21

MSR Address: F01h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.278 (F02h) PERF_UNIT_CTL_1_CHA_21



MSR Address: F02h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.279 (F03h) PERF_UNIT_STATUS_CHA_21

Cbo Pmon counters status

MSR Address: F03h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.280 (F04h) PERF_CTR_0_CHA_21

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F04h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.281 (F05h) PERF_CTR_1_CHA_21

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F05h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.282 (F06h) PERF_CTR_2_CHA_21

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F06h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.283 (F07h) PERF_CTR_3_CHA_21

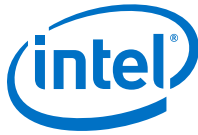
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F07h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.284 (F08h) PERF_CTR_UNIT_CTRL_CHA_22

Pmon Unit control.

MSR Address: F08h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
<i>continued...</i>				



MSR Address: F08h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.285 (F09h) PERF_EVT_SEL_0_CHA_22

Perfmon Counter Control Register

MSR Address: F09h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				



MSR Address: F09h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.286 (F0Ah) PERF_EVT_SEL_1_CHA_22

Perfmon Counter Control Register

MSR Address: F0Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: F0Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

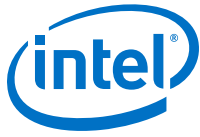
5.2.287 (F0Bh) PERF_EVT_SEL_2_CHA_22

Perfmon Counter Control Register



MSR Address: F0Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: F0Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.288 (F0Ch) PERF_EVT_SEL_3_CHA_22

Perfmon Counter Control Register

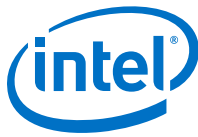
MSR Address: F0Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: F0Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.289 (F0Dh) PERF_UNIT_CTL_CHA_22

MSR Address: F0Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F0Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.290 (F0Eh) PERF_UNIT_CTL_1_CHA_22

MSR Address: F0Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.291 (F0Fh) PERF_UNIT_STATUS_CHA_22

Cbo Pmon counters status

MSR Address: F0Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
continued...				



MSR Address: F0Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.292 (F10h) PERF_CTR_0_CHA_22

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F10h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.293 (F11h) PERF_CTR_1_CHA_22

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F11h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.294 (F12h) PERF_CTR_2_CHA_22

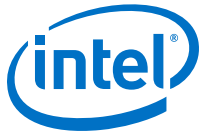
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F12h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.295 (F13h) PERF_CTR_3_CHA_22

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F13h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.296 (F14h) PERF_CTR_UNIT_CTRL_CHA_23

Pmon Unit control.

MSR Address: F14h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.297 (F15h) PERF_EVT_SEL_0_CHA_23

Perfmon Counter Control Register

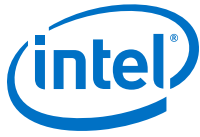
MSR Address: F15h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: F15h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.298 (F16h) PERF_EVT_SEL_1_CHA_23

Perfmon Counter Control Register



MSR Address: F16h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

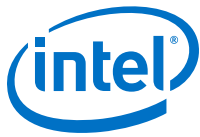


MSR Address: F16h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.299 (F17h) PERF_EVT_SEL_2_CHA_23

Perfmon Counter Control Register

MSR Address: F17h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: F17h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.300 (F18h) PERF_EVT_SEL_3_CHA_23

Perfmon Counter Control Register

MSR Address: F18h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				

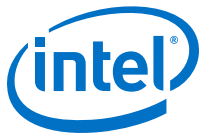


MSR Address: F18h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.301 (F19h) PERF_UNIT_CTL_CHA_23

MSR Address: F19h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved

continued...



MSR Address: F19h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.302 (F1Ah) PERF_UNIT_CTL_1_CHA_23

MSR Address: F1Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.303 (F1Bh) PERF_UNIT_STATUS_CHA_23

Cbo Pmon counters status

MSR Address: F1Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.304 (F1Ch) PERF_CTR_0_CHA_23

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F1Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.305 (F1Dh) PERF_CTR_1_CHA_23

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F1Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

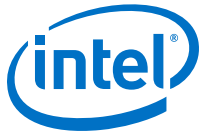
5.2.306 (F1Eh) PERF_CTR_2_CHA_23

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F1Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.307 (F1Fh) PERF_CTR_3_CHA_23

This register is a perfmon counter. Software can both read it and write it.



MSR Address: F1Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.308 (F20h) PERF_CTR_UNIT_CTRL_CHA_24

Pmon Unit control.

MSR Address: F20h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.309 (F21h) PERF_EVT_SEL_0_CHA_24

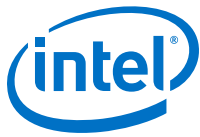
Perfmon Counter Control Register

MSR Address: F21h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: F21h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected

continued...



MSR Address: F21h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.310 (F22h) PERF_EVT_SEL_1_CHA_24

Perfmon Counter Control Register

MSR Address: F22h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.
continued...				

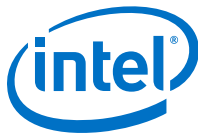


MSR Address: F22h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.311 (F23h) PERF_EVT_SEL_2_CHA_24

Perfmon Counter Control Register

MSR Address: F23h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F23h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

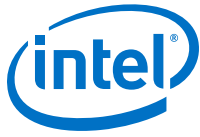
5.2.312 (F24h) PERF_EVT_SEL_3_CHA_24

Perfmon Counter Control Register

MSR Address: F24h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: F24h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.313 (F25h) PERF_UNIT_CTL_CHA_24

MSR Address: F25h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.314 (F26h) PERF_UNIT_CTL_1_CHA_24

MSR Address: F26h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: F26h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.315 (F27h) PERF_UNIT_STATUS_CHA_24

Cbo Pmon counters status

MSR Address: F27h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.316 (F28h) PERF_CTR_0_CHA_24

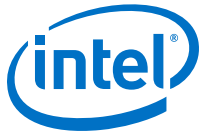
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F28h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.317 (F29h) PERF_CTR_1_CHA_24

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F29h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.318 (F2Ah) PERF_CTR_2_CHA_24

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F2Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.319 (F2Bh) PERF_CTR_3_CHA_24

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F2Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.320 (F2Ch) PERF_CTR_UNIT_CTRL_CHA_25

Pmon Unit control.

MSR Address: F2Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				

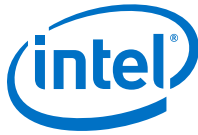


MSR Address: F2Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.321 (F2Dh) PERF_EVT_SEL_0_CHA_25

Perfmon Counter Control Register

MSR Address: F2Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				



MSR Address: F2Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.322 (F2Eh) PERF_EVT_SEL_1_CHA_25

Perfmon Counter Control Register

MSR Address: F2Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				

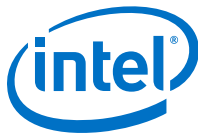


MSR Address: F2Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.323 (F2Fh) PERF_EVT_SEL_2_CHA_25

Perfmon Counter Control Register

MSR Address: F2Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: F2Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

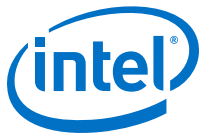


5.2.324 (F30h) PERF_EVT_SEL_3_CHA_25

Perfmon Counter Control Register

MSR Address: F30h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: F30h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.325 (F31h) PERF_UNIT_CTL_CHA_25

MSR Address: F31h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.326 (F32h) PERF_UNIT_CTL_1_CHA_25



MSR Address: F32h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.327 (F33h) PERF_UNIT_STATUS_CHA_25

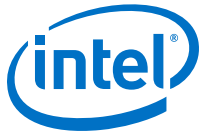
Cbo Pmon counters status

MSR Address: F33h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.328 (F34h) PERF_CTR_0_CHA_25

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F34h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.329 (F35h) PERF_CTR_1_CHA_25

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F35h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.330 (F36h) PERF_CTR_2_CHA_25

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F36h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.331 (F37h) PERF_CTR_3_CHA_25

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F37h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.332 (F38h) PERF_CTR_UNIT_CTRL_CHA_26

Pmon Unit control.

MSR Address: F38h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
continued...				

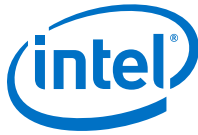


MSR Address: F38h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.333 (F39h) PERF_EVT_SEL_0_CHA_26

Perfmon Counter Control Register

MSR Address: F39h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				



MSR Address: F39h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.334 (F3Ah) PERF_EVT_SEL_1_CHA_26

Perfmon Counter Control Register

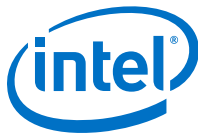
MSR Address: F3Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: F3Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.335 (F3Bh) PERF_EVT_SEL_2_CHA_26

Perfmon Counter Control Register



MSR Address: F3Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

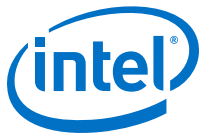


MSR Address: F3Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.336 (F3Ch) PERF_EVT_SEL_3_CHA_26

Perfmon Counter Control Register

MSR Address: F3Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: F3Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.337 (F3Dh) PERF_UNIT_CTL_CHA_26

MSR Address: F3Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F3Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.338 (F3Eh) PERF_UNIT_CTL_1_CHA_26

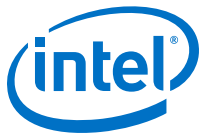
MSR Address: F3Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.339 (F3Fh) PERF_UNIT_STATUS_CHA_26

Cbo Pmon counters status

MSR Address: F3Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed

continued...



MSR Address: F3Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.340 (F40h) PERF_CTR_0_CHA_26

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F40h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.341 (F41h) PERF_CTR_1_CHA_26

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F41h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.342 (F42h) PERF_CTR_2_CHA_26

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F42h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.343 (F43h) PERF_CTR_3_CHA_26

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F43h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.344 (F44h) PERF_CTR_UNIT_CTRL_CHA_27

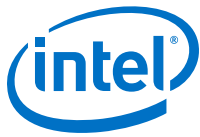
Pmon Unit control.

MSR Address: F44h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.345 (F45h) PERF_EVT_SEL_0_CHA_27

Perfmon Counter Control Register

MSR Address: F45h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: F45h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

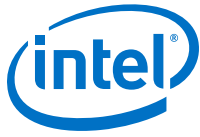
5.2.346 (F46h) PERF_EVT_SEL_1_CHA_27

Perfmon Counter Control Register



MSR Address: F46h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: F46h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.347 (F47h) PERF_EVT_SEL_2_CHA_27

Perfmon Counter Control Register

MSR Address: F47h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

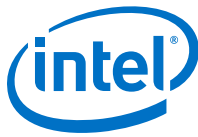


MSR Address: F47h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.348 (F48h) PERF_EVT_SEL_3_CHA_27

Perfmon Counter Control Register

MSR Address: F48h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: F48h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.349 (F49h) PERF_UNIT_CTL_CHA_27

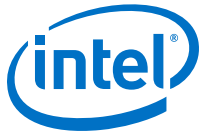
MSR Address: F49h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved
continued...				



MSR Address: F49h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.350 (F4Ah) PERF_UNIT_CTL_1_CHA_27

MSR Address: F4Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.351 (F4Bh) PERF_UNIT_STATUS_CHA_27

Cbo Pmon counters status

MSR Address: F4Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.352 (F4Ch) PERF_CTR_0_CHA_27

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F4Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.353 (F4Dh) PERF_CTR_1_CHA_27

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F4Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

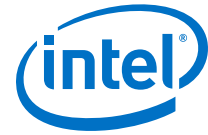
5.2.354 (F4Eh) PERF_CTR_2_CHA_27

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F4Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.355 (F4Fh) PERF_CTR_3_CHA_27

This register is a perfmon counter. Software can both read it and write it.



MSR Address: F4Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.356 (F50h) PERF_CTR_UNIT_CTRL_CHA_28

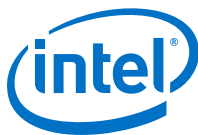
Pmon Unit control.

MSR Address: F50h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.357 (F51h) PERF_EVT_SEL_0_CHA_28

Perfmon Counter Control Register

MSR Address: F51h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: F51h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				



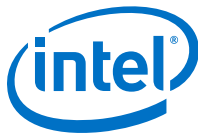
MSR Address: F51h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.358 (F52h) PERF_EVT_SEL_1_CHA_28

Perfmon Counter Control Register

MSR Address: F52h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.

continued...



MSR Address: F52h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.359 (F53h) PERF_EVT_SEL_2_CHA_28

Perfmon Counter Control Register

MSR Address: F53h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				

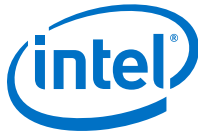


MSR Address: F53h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.360 (F54h) PERF_EVT_SEL_3_CHA_28

Perfmon Counter Control Register

MSR Address: F54h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: F54h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

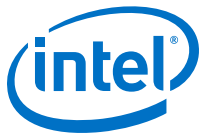


5.2.361 (F55h) PERF_UNIT_CTL_CHA_28

MSR Address: F55h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.362 (F56h) PERF_UNIT_CTL_1_CHA_28

MSR Address: F56h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: F56h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.363 (F57h) PERF_UNIT_STATUS_CHA_28

Cbo Pmon counters status

MSR Address: F57h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.364 (F58h) PERF_CTR_0_CHA_28

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F58h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.365 (F59h) PERF_CTR_1_CHA_28

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F59h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.366 (F5Ah) PERF_CTR_2_CHA_28

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F5Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.367 (F5Bh) PERF_CTR_3_CHA_28

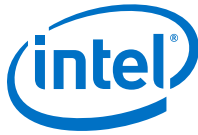
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F5Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.368 (F5Ch) PERF_CTR_UNIT_CTRL_CHA_29

Pmon Unit control.

MSR Address: F5Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				



MSR Address: F5Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.369 (F5Dh) PERF_EVT_SEL_0_CHA_29

Perfmon Counter Control Register

MSR Address: F5Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				

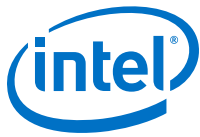


MSR Address: F5Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.370 (F5Eh) PERF_EVT_SEL_1_CHA_29

Perfmon Counter Control Register

MSR Address: F5Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F5Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

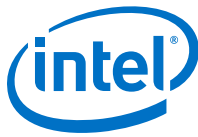
5.2.371 (F5Fh) PERF_EVT_SEL_2_CHA_29

Perfmon Counter Control Register

MSR Address: F5Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: F5Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

**5.2.372 (F60h) PERF_EVT_SEL_3_CHA_29**

Perfmon Counter Control Register

MSR Address: F60h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				



MSR Address: F60h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

5.2.373 (F61h) PERF_UNIT_CTL_CHA_29

MSR Address: F61h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.374 (F62h) PERF_UNIT_CTL_1_CHA_29



MSR Address: F62h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.375 (F63h) PERF_UNIT_STATUS_CHA_29

Cbo Pmon counters status

MSR Address: F63h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.376 (F64h) PERF_CTR_0_CHA_29

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F64h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.377 (F65h) PERF_CTR_1_CHA_29

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F65h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.378 (F66h) PERF_CTR_2_CHA_29

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F66h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.379 (F67h) PERF_CTR_3_CHA_29

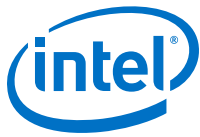
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F67h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.380 (F68h) PERF_CTR_UNIT_CTRL_CHA_30

Pmon Unit control.

MSR Address: F68h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
<i>continued...</i>				



MSR Address: F68h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.381 (F69h) PERF_EVT_SEL_0_CHA_30

Perfmon Counter Control Register

MSR Address: F69h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				

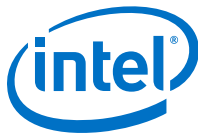


MSR Address: F69h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.382 (F6Ah) PERF_EVT_SEL_1_CHA_30

Perfmon Counter Control Register

MSR Address: F6Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: F6Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

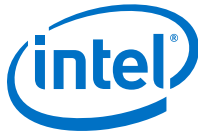
5.2.383 (F6Bh) PERF_EVT_SEL_2_CHA_30

Perfmon Counter Control Register



MSR Address: F6Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: F6Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.384 (F6Ch) PERF_EVT_SEL_3_CHA_30

Perfmon Counter Control Register

MSR Address: F6Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

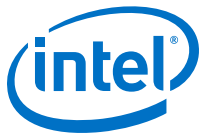


MSR Address: F6Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.385 (F6Dh) PERF_UNIT_CTL_CHA_30

MSR Address: F6Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.

continued...



MSR Address: F6Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.386 (F6Eh) PERF_UNIT_CTL_1_CHA_30

MSR Address: F6Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.387 (F6Fh) PERF_UNIT_STATUS_CHA_30

Cbo Pmon counters status

MSR Address: F6Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
continued...				



MSR Address: F6Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.388 (F70h) PERF_CTR_0_CHA_30

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F70h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.389 (F71h) PERF_CTR_1_CHA_30

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F71h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.390 (F72h) PERF_CTR_2_CHA_30

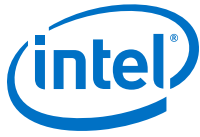
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F72h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.391 (F73h) PERF_CTR_3_CHA_30

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F73h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.392 (F74h) PERF_CTR_UNIT_CTRL_CHA_31

Pmon Unit control.

MSR Address: F74h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.393 (F75h) PERF_EVT_SEL_0_CHA_31

Perfmon Counter Control Register

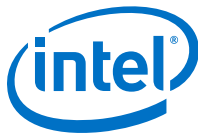
MSR Address: F75h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: F75h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.394 (F76h) PERF_EVT_SEL_1_CHA_31

Perfmon Counter Control Register



MSR Address: F76h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...

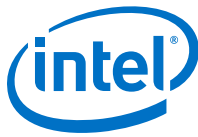


MSR Address: F76h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.395 (F77h) PERF_EVT_SEL_2_CHA_31

Perfmon Counter Control Register

MSR Address: F77h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: F77h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.396 (F78h) PERF_EVT_SEL_3_CHA_31

Perfmon Counter Control Register

MSR Address: F78h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				

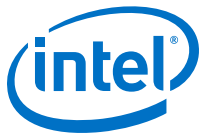


MSR Address: F78h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.397 (F79h) PERF_UNIT_CTL_CHA_31

MSR Address: F79h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved

continued...



MSR Address: F79h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.398 (F7Ah) PERF_UNIT_CTL_1_CHA_31

MSR Address: F7Ah				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.399 (F7Bh) PERF_UNIT_STATUS_CHA_31

Cbo Pmon counters status

MSR Address: F7Bh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.400 (F7Ch) PERF_CTR_0_CHA_31

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F7Ch				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.401 (F7Dh) PERF_CTR_1_CHA_31

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F7Dh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.402 (F7Eh) PERF_CTR_2_CHA_31

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F7Eh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.403 (F7Fh) PERF_CTR_3_CHA_31

This register is a perfmon counter. Software can both read it and write it.



MSR Address: F7Fh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.404 (F80h) PERF_CTR_UNIT_CTRL_CHA_32

Pmon Unit control.

MSR Address: F80h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.405 (F81h) PERF_EVT_SEL_0_CHA_32

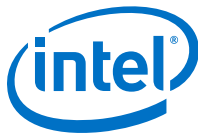
Perfmon Counter Control Register

MSR Address: F81h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: F81h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected

continued...



MSR Address: F81h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.406 (F82h) PERF_EVT_SEL_1_CHA_32

Perfmon Counter Control Register

MSR Address: F82h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.
continued...				

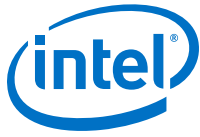


MSR Address: F82h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.407 (F83h) PERF_EVT_SEL_2_CHA_32

Perfmon Counter Control Register

MSR Address: F83h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F83h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.408 (F84h) PERF_EVT_SEL_3_CHA_32

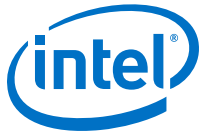
Perfmon Counter Control Register

MSR Address: F84h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see

continued...



MSR Address: F84h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.



5.2.409 (F85h) PERF_UNIT_CTL_CHA_32

MSR Address: F85h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.410 (F86h) PERF_UNIT_CTL_1_CHA_32

MSR Address: F86h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: F86h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.411 (F87h) PERF_UNIT_STATUS_CHA_32

Cbo Pmon counters status

MSR Address: F87h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.412 (F88h) PERF_CTR_0_CHA_32

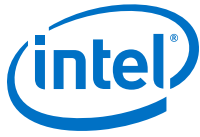
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F88h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.413 (F89h) PERF_CTR_1_CHA_32

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F89h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.414 (F8Ah) PERF_CTR_2_CHA_32

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F8Ah				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.415 (F8Bh) PERF_CTR_3_CHA_32

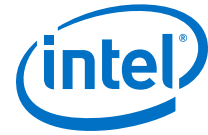
This register is a perfmon counter. Software can both read it and write it.

MSR Address: F8Bh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.416 (F8Ch) PERF_CTR_UNIT_CTRL_CHA_33

Pmon Unit control.

MSR Address: F8Ch				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				

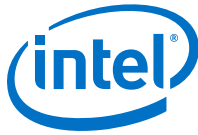


MSR Address: F8Ch				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.417 (F8Dh) PERF_EVT_SEL_0_CHA_33

Perfmon Counter Control Register

MSR Address: F8Dh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				



MSR Address: F8Dh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.418 (F8Eh) PERF_EVT_SEL_1_CHA_33

Perfmon Counter Control Register

MSR Address: F8Eh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				

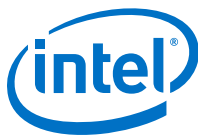


MSR Address: F8Eh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.419 (F8Fh) PERF_EVT_SEL_2_CHA_33

Perfmon Counter Control Register

MSR Address: F8Fh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: F8Fh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold \leq event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\event select\q, \unit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

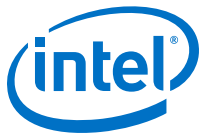


5.2.420 (F90h) PERF_EVT_SEL_3_CHA_33

Perfmon Counter Control Register

MSR Address: F90h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: F90h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.421 (F91h) PERF_UNIT_CTL_CHA_33

MSR Address: F91h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.422 (F92h) PERF_UNIT_CTL_1_CHA_33



MSR Address: F92h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.423 (F93h) PERF_UNIT_STATUS_CHA_33

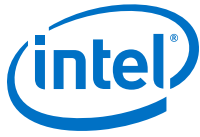
Cbo Pmon counters status

MSR Address: F93h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.424 (F94h) PERF_CTR_0_CHA_33

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F94h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.425 (F95h) PERF_CTR_1_CHA_33

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F95h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.426 (F96h) PERF_CTR_2_CHA_33

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F96h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.427 (F97h) PERF_CTR_3_CHA_33

This register is a perfmon counter. Software can both read it and write it.

MSR Address: F97h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.428 (F98h) PERF_CTR_UNIT_CTRL_CHA_34

Pmon Unit control.

MSR Address: F98h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
continued...				



MSR Address: F98h				
Bit	Scope	Default	Attribute	Description
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.429 (F99h) PERF_EVT_SEL_0_CHA_34

Perfmon Counter Control Register

MSR Address: F99h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
continued...				



MSR Address: F99h				
Bit	Scope	Default	Attribute	Description
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.430 (F9Ah) PERF_EVT_SEL_1_CHA_34

Perfmon Counter Control Register

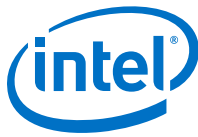
MSR Address: F9Ah				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
continued...				



MSR Address: F9Ah				
Bit	Scope	Default	Attribute	Description
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would selet the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.431 (F9Bh) PERF_EVT_SEL_2_CHA_34

Perfmon Counter Control Register



MSR Address: F9Bh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				

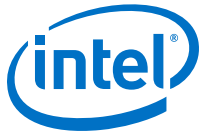


MSR Address: F9Bh				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.432 (F9Ch) PERF_EVT_SEL_3_CHA_34

Perfmon Counter Control Register

MSR Address: F9Ch				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				



MSR Address: F9Ch				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drops to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.433 (F9Dh) PERF_UNIT_CTL_CHA_34

MSR Address: F9Dh				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: F9Dh				
Bit	Scope	Default	Attribute	Description
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.434 (F9Eh) PERF_UNIT_CTL_1_CHA_34

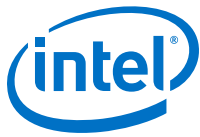
MSR Address: F9Eh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.435 (F9Fh) PERF_UNIT_STATUS_CHA_34

Cbo Pmon counters status

MSR Address: F9Fh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed

continued...



MSR Address: F9Fh				
Bit	Scope	Default	Attribute	Description
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.436 (FA0h) PERF_CTR_0_CHA_34

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FA0h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.437 (FA1h) PERF_CTR_1_CHA_34

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FA1h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.438 (FA2h) PERF_CTR_2_CHA_34

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FA2h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.439 (FA3h) PERF_CTR_3_CHA_34

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FA3h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.440 (FA4h) PERF_CTR_UNIT_CTRL_CHA_35

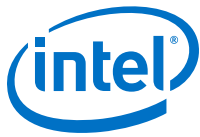
Pmon Unit control.

MSR Address: FA4h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.441 (FA5h) PERF_EVT_SEL_0_CHA_35

Perfmon Counter Control Register

MSR Address: FA5h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event.
continued...				



MSR Address: FA5h				
Bit	Scope	Default	Attribute	Description
				event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

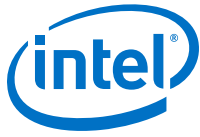
5.2.442 (FA6h) PERF_EVT_SEL_1_CHA_35

Perfmon Counter Control Register



MSR Address: FA6h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is

continued...



MSR Address: FA6h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.443 (FA7h) PERF_EVT_SEL_2_CHA_35

Perfmon Counter Control Register

MSR Address: FA7h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true
continued...				

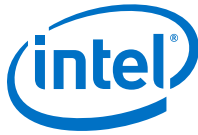


MSR Address: FA7h				
Bit	Scope	Default	Attribute	Description
				even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlect) — This field is used to decode the PerfMon event which is selected.

5.2.444 (FA8h) PERF_EVT_SEL_3_CHA_35

Perfmon Counter Control Register

MSR Address: FA8h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
continued...				



MSR Address: FA8h				
Bit	Scope	Default	Attribute	Description
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.445 (FA9h) PERF_UNIT_CTL_CHA_35

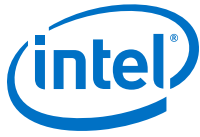
MSR Address: FA9h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved
continued...				



MSR Address: FA9h				
Bit	Scope	Default	Attribute	Description
				[18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.446 (FAAh) PERF_UNIT_CTL_1_CHA_35

MSR Address: FAAh				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.



5.2.447 (FABh) PERF_UNIT_STATUS_CHA_35

Cbo Pmon counters status

MSR Address: FABh				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.448 (FACH) PERF_CTR_0_CHA_35

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FACH				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.449 (FADh) PERF_CTR_1_CHA_35

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FADh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.450 (FAEh) PERF_CTR_2_CHA_35

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FAEh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.451 (FAFh) PERF_CTR_3_CHA_35

This register is a perfmon counter. Software can both read it and write it.



MSR Address: FAFh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.452 (FB0h) PERF_CTR_UNIT_CTRL_CHA_36

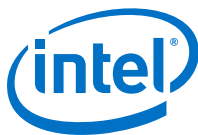
Pmon Unit control.

MSR Address: FB0h				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.453 (FB1h) PERF_EVT_SEL_0_CHA_36

Perfmon Counter Control Register

MSR Address: FB1h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see <i>continued...</i>)



MSR Address: FB1h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold \neq 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the <code>\qevent select\q, \qunit mask\q</code> . There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a <code>\q1\q</code> is written to it. No action is taken when a <code>\q0\q</code> is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected
continued...				



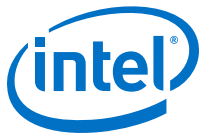
MSR Address: FB1h				
Bit	Scope	Default	Attribute	Description
				otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.454 (FB2h) PERF_EVT_SEL_1_CHA_36

Perfmon Counter Control Register

MSR Address: FB2h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.

continued...



MSR Address: FB2h				
Bit	Scope	Default	Attribute	Description
				Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.455 (FB3h) PERF_EVT_SEL_2_CHA_36

Perfmon Counter Control Register

MSR Address: FB3h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: FB3h				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

5.2.456 (FB4h) PERF_EVT_SEL_3_CHA_36

Perfmon Counter Control Register

MSR Address: FB4h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: FB4h				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \event select\q, \unit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlt) — This field is used to decode the PerfMon event which is selected.

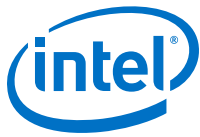


5.2.457 (FB5h) PERF_UNIT_CTL_CHA_36

MSR Address: FB5h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.458 (FB6h) PERF_UNIT_CTL_1_CHA_36

MSR Address: FB6h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
continued...				



MSR Address: FB6h				
Bit	Scope	Default	Attribute	Description
3	package	0h	RW	Count for all opcodes (AllOpCodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.459 (FB7h) PERF_UNIT_STATUS_CHA_36

Cbo Pmon counters status

MSR Address: FB7h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.460 (FB8h) PERF_CTR_0_CHA_36

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FB8h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.461 (FB9h) PERF_CTR_1_CHA_36

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FB9h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.462 (FBAh) PERF_CTR_2_CHA_36

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FBAh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.463 (FBBh) PERF_CTR_3_CHA_36

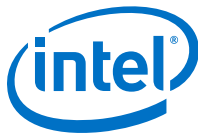
This register is a perfmon counter. Software can both read it and write it.

MSR Address: FBBh				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.464 (FBCh) PERF_CTR_UNIT_CTRL_CHA_37

Pmon Unit control.

MSR Address: FBCh				
Bit	Scope	Default	Attribute	Description
31:18	-	-	-	Reserved (RSVD) — Reserved.
17	package	1h	RW_L	Overflow Enable (OverflowEnable) — This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count. For overflow to be enabled for a given unit, all of the unit control registers must have this bit set.
16	package	1h	RW_L	Freeze Enable (FreezeEnable) — This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal. For freeze to be enabled for a given unit, all of the unit control registers must have this bit set.
15:9	-	-	-	Reserved (RSVD) — Reserved.
8	package	0h	RW_V	Freeze Counters (FreezeCounters) — This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting. To freeze the counters, this bit need only be set by one of the unit control registers.
continued...				



MSR Address: FBCh				
Bit	Scope	Default	Attribute	Description
7:2	-	-	-	Reserved (RSVD) — Reserved.
1	package	0h	WO	Reset Counters (ResetCounters) — When this bit is written to, the counters data fields will be reset. The configuration values will not be reset. To reset the counters, this bit need only be set by one of the unit control registers.
0	package	0h	WO	Reset Counter Configs (ResetCounterConfigs) — When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters. To reset the counters, this bit need only be set by one of the unit control registers.

5.2.465 (FBDh) PERF_EVT_SEL_0_CHA_37

Perfmon Counter Control Register

MSR Address: FBDh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1.
continued...				

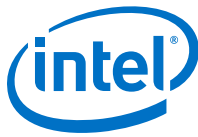


MSR Address: FBDh				
Bit	Scope	Default	Attribute	Description
				One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

5.2.466 (FBEh) PERF_EVT_SEL_1_CHA_37

Perfmon Counter Control Register

MSR Address: FBEh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
continued...				



MSR Address: FBEh				
Bit	Scope	Default	Attribute	Description
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

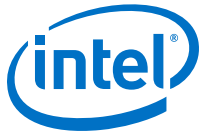
5.2.467 (FBFh) PERF_EVT_SEL_2_CHA_37

Perfmon Counter Control Register

MSR Address: FBFh				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see
continued...				



MSR Address: FBFh				
Bit	Scope	Default	Attribute	Description
				bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always on differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSict) — This field is used to decode the PerfMon event which is selected.

**5.2.468 (FC0h) PERF_EVT_SEL_3_CHA_37**

Perfmon Counter Control Register

MSR Address: FC0h				
Bit	Scope	Default	Attribute	Description
31:24	package	0h	RW	Threshold — This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	package	0h	RW	Invert — This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	package	0h	RW	Counter Enable (CounterEn) — This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the \qevent select\q, \qunit mask\q. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
20	package	0h	RW	Overflow Enable (OvfEnable) — Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
19	package	0h	RW	ThreadID Filter Enable (TidFilterEnable) — ThradID filter enable. This is only used by Cbo. For other units it is Reserved.
18	package	0h	RW	Edge Detect (EdgeDet) — Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	package	0h	WO	Counter Reset (CounterReset) — When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	package	0h	WO	Queue Occupancy Reset (QueueOccupancyReset) — This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a \q1\q is written to it. No action is taken when a \q0\q is
continued...				

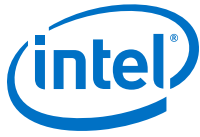


MSR Address: FC0h				
Bit	Scope	Default	Attribute	Description
				written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	package	0h	RW	Unit Mask (UnitMask) — This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	package	0h	RW	Event Select (EvSlct) — This field is used to decode the PerfMon event which is selected.

5.2.469 (FC1h) PERF_UNIT_CTL_CHA_37

MSR Address: FC1h				
Bit	Scope	Default	Attribute	Description
31:27	-	-	-	Reserved (RSVD) — Reserved.
26:17	package	0h	RW	State — Select the state(s) to monitor in the CBO_CACHE_LOOKUP event (when the STATE submask is selected). Setting multiple bits in this field will allow one to track multiple states. [17] - Reserved [18] - SF_S [19] - SF_E [20] - SF_H [21] - Reserved [22] - Reserved [23] - Reserved [24] - Reserved [25] - Reserved [26] - Reserved
16:13	-	-	-	Reserved (RSVD) — Reserved.
12	package	0h	RW	Link3 — Selects whether M2PCIE credits will be monitored in the KTI_IGR_CREDIT_ALLOC and OCCUPANCY events.
11	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
10	package	0h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
9	package	1h	RSVD-P	Reserved (RSVD-P) — Reserved - protected.
8:0	package	0h	RW	Thread-ID (ThreadId) — Thread-ID format:[2:0] - ThreadId. [8:3] - Core-ID

5.2.470 (FC2h) PERF_UNIT_CTL_1_CHA_37



MSR Address: FC2h				
Bit	Scope	Default	Attribute	Description
31	package	0h	RW	IsocOpcode — 0 for all requests, 1 for ISOC requests
30	package	0h	RW	NonCohOpcode — 0 for all requests, 1 for non-coh requests
29	package	0h	RW	C6Opcode — 0 for all requests, 1 for C6 requests
28:19	package	0h	RW	Opcode (Opcode1) — Opcode1: The Second opcode value for the opcode based filtering.
18:9	package	0h	RW	Opcode (Opcode0) — Opcode0: The first opcode value for the opcode based filtering.
8:6	-	-	-	Reserved (RSVD) — Reserved.
5	package	1h	RW	Count non-near memory cache events (NotNearMemCacheable) — Reserved.
4	package	1h	RW	Count near memory cache events (NearMemCacheable) — Reserved.
3	package	0h	RW	Count for all opcodes (AllOpcodes) — If set to 1, disables the Opcode filters and instead causes the TOR_ALLOC and TOR_OCCUPANCY events to match on all opcodes. If set to 0, then requests then the TOR_OCCUPANCY and TOR_ALLOC events will match only when a request has an opcode that matches the value in either Opcode0 or Opcode1.
2	-	-	-	Reserved (RSVD) — Reserved.
1	package	1h	RW	Local Node (LocalNode) — Reserved.
0	package	0h	RW	Target Node (RemoteNode) — Reserved.

5.2.471 (FC3h) PERF_UNIT_STATUS_CHA_37

Cbo Pmon counters status

MSR Address: FC3h				
Bit	Scope	Default	Attribute	Description
31:4	-	-	-	Reserved (RSVD) — Reserved.
3	package	0h	RW1C	Counter 3 Overflowed (Counter3Ovf) — counter 3 overflowed
2	package	0h	RW1C	Counter 2 Overflowed (Counter2Ovf) — counter 2 overflowed
1	package	0h	RW1C	Counter 1 Overflowed (Counter1Ovf) — counter 1 overflowed
0	package	0h	RW1C	Counter 0 Overflowed (Counter0Ovf) — counter 0 overflowed

5.2.472 (FC4h) PERF_CTR_0_CHA_37

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FC4h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



5.2.473 (FC5h) PERF_CTR_1_CHA_37

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FC5h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.474 (FC6h) PERF_CTR_2_CHA_37

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FC6h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.

5.2.475 (FC7h) PERF_CTR_3_CHA_37

This register is a perfmon counter. Software can both read it and write it.

MSR Address: FC7h				
Bit	Scope	Default	Attribute	Description
63:48	-	-	-	Reserved (RSVD) — Reserved.
47:0	package	0h	RW_V	Counter Value (PmonCtrData) — This is the current value of the counter.



6.0 IRP

Table 38. Summary of Bus:0 Device:5 Function:6

Offset	Register ID
A0h	(A0h)IRP0_PMONCNTR_0: PMON Counter
A8h	(A8h)IRP0_PMONCNTR_1: PMON Counter
D8h	(D8h)IRP0_PMONCNTRCFG_0: Counter Config
DCh	(DCh)IRP0_PMONCNTRCFG_1: Counter Config
F0h	(F0h)IRP_PMONUNITCTRL: Unit Control
F4h	(F4h)IRP_PMONUNITSTATUS: Unit Status

6.1 IRP0_PMONCNTR_0: PMON Counter

This register is a perfmon counter. Software can both read it and write it.

Bus: RootBus0 Device: 5 Function: 6 Offset : A0h			
Bit	Attribute	Default	Description
63:48	RV	0	Reserved
47:0	RW-V	0	CounterValue -This is the current value of the counter.

6.2 IRP0_PMONCNTR_1: PMON Counter

This register is a perfmon counter. Software can both read it and write it.

Bus: RootBus0 Device: 5 Function: 6 Offset : A8h			
Bit	Attribute	Default	Description
63:48	RV	0	Reserved
47:0	RW-V	0	CounterValue -This is the current value of the counter.

6.3 IRP0_PMONCNTRCFG_0: Counter Config

Perfmon Counter Control Register

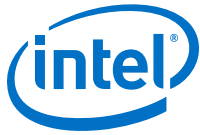
Bus: RootBus0 Device: 5 Function: 6 Offset : D8h			
Bit	Attribute	Default	Description
31:24	RW-V	0b	Threshold This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison

continued...



Bus: RootBus0 Device: 5 Function: 6 Offset : D8h			
Bit	Attribute	Default	Description
			depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	RW-V	0b	<p>Invert</p> <p>This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold \geq event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold $<$ event. The invert bit only works when Threshold $\neq 0$. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.</p>
22	RW-V	0b	<p>CounterEnable</p> <p>This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.</p>
21	-	-	Reserved (RSVD) - Reserved
20	RW-V	0b	<p>OverflowEnable</p> <p>Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.</p>
18	RW-V	0b	<p>EdgeDetect</p> <p>Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge).</p> <p>Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit.</p> <p>Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.</p>
17	WO	0b	<p>CounterReset</p> <p>When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.</p>
16	WO	0b	<p>QueueOccupancyReset</p> <p>This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a '1' is written to it. No action is taken when a '0' is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.</p>
15:8	RW-V	00h	<p>UnitMask</p> <p>This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.</p>
7:0	RW-V	00h	EventSelect

continued...



Bus: RootBus0 Device: 5 Function: 6 Offset : D8h			
Bit	Attribute	Default	Description
			This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective individual unit performance monitoring documentation.

6.4 IRP0_PMONCNTRCFG_1: Counter Config

Perfmon Counter Control Register

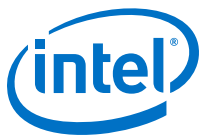
Bus: RootBus0 Device: 5 Function: 6 Offset : DCh			
Bit	Attribute	Default	Description
31:24	RW-V	0b	Threshold This field is compared directly against an incoming event value for events that can increment by 1 or more in a given cycle. Since the widest event from the UnCore is 7bits (queue occupancy), bit 31 is unused. The result of the comparison is effectively a 1 bit wide event, i.e., the counter will be incremented by 1 when the comparison is true (the type of comparison depends on the setting of the 'invert' bit - see bit 23 below) no matter how wide the original event was. When this field is zero, threshold comparison is disabled and the event is passed without modification.
23	RW-V	0b	Invert This bit indicates how the threshold field will be compared to the incoming event. When 0, the comparison that will be done is threshold >= event. When set to 1, the comparison that will be done is inverted from the case where this bit is set to 0, i.e., threshold < event. The invert bit only works when Threshold != 0. So, if one would like to invert a non-occupancy event (like LLC Hit), one needs to set the threshold to 1.
22	RW-V	0b	CounterEnable This field is the local enable for the PerfMon Counter. This bit must be asserted in order for the PerfMon counter to begin counting the events selected by the event select and unit mask. There is one bit per PerfMon Counter. Note that if this bit is set to 1 but the Unit Control Registers have determined that counting is disabled, then the counter will not count.
21	-	-	Reserved (RSVD) - Reserved
20	RW-V	0b	OverflowEnable Setting this bit will enable the counter to send an overflow signal. If this bit is not set, the counter will wrap around when it overflows without triggering anything. If this bit is set and the Unit's configuration register has Overflow enabled, then a signal will be transmitted to the Ubox.
18	RW-V	0b	EdgeDetect Edge Detect allows one to count either 0 to 1 or 1 to 0 transitions of a given event. For example, we have an event that counts the number of cycles in L0s mode in QPI. By using edge detect, one can count the number of times that we entered L0s mode (by detecting the rising edge). Edge detect only works in conjunction with thresholding. This is true even for events that can only increment by 1 in a given cycle (like the L0s example above). In this case, one should set a threshold of 1. One can also use Edge Detect with queue occupancy events. For example, if one wanted to count the number of times when the TOR occupancy was larger than 5, one would select the TOR occupancy event with a threshold of 5 and set the Edge Detect bit. Edge detect can also be used with the invert. This is generally not particularly useful, as the count of falling edges compared to rising edges will always differ by 1.
17	WO	0b	CounterReset
continued...			



Bus: RootBus0 Device: 5 Function: 6 Offset : DCh			
Bit	Attribute	Default	Description
			When this bit is set, the corresponding counter will be reset to 0. This allows for a quick reset of the counter when changing event encodings.
16	WO	0b	QueueOccupancyReset This write only bit causes the queue occupancy counter of the PerfMon counter for which this Perf event select register is associated to be cleared to all zeroes when a '1' is written to it. No action is taken when a '0' is written. Note: Since the queue occupancy counters never drop below zero, it is possible for the counters to 'catch up' with the real occupancy of the queue in question when the real occupancy drop to zero.
15:8	RW-V	00h	UnitMask This mask selects the sub-events to be selected for creation of the event. The selected sub-events are bitwise OR-ed together to create event. At least one sub-event must be selected otherwise the PerfMon event signals will not ever get asserted. Events with no sub-events listed effectively have only one sub-event -- bit 8 must be set to 1 in this case.
7:0	RW-V	00h	EventSelect This field is used to decode the PerfMon event which is selected. The encodings for each of the valid UnCore PerfMon events can be found in the respective individual unit performance monitoring documentation.

6.5 IRP_PMONUNITCTRL: Unit Control

Bus: RootBus0 Device: 5 Function: 6 Offset : F0h			
Bit	Attribute	Default	Description
31:19	RV	0	Reserved
18	RW	0	1
17	RO	1	OverflowEnable This bit controls the behavior of counters when they overflow. When set, the system will trigger the overflow handling process throughout the rest of the uncore, potentially triggering a PMI and freezing counters. When it is not set, the counters will simply wrap around and continue to count.
16	RO	0b	FreezeEnable This bit controls what the counters in the unit will do when they receive a freeze signal. When set, the counters will be allowed to freeze. When not set, the counters will ignore the freeze signal.
15:9	RV	0	Reserved
8	RW	0b	FreezeCounters This bit is written to when the counters should be frozen. If this bit is written to and freeze is enabled, the counters in the unit will stop counting.
7:2	RV	0	Reserved
1	WO	0b	ResetCounters When this bit is written to, the counters data fields will be reset. The configuration values will not be reset.
0	WO	0b	ResetCounterConfigs When this bit is written to, the counter configuration registers will be reset. This does not effect the values in the counters.



6.6 IRP_PMONUNITSTATUS: Unit Status

This field shows which registers have overflowed in the unit.

Whenever a register overflows, it should set the relevant bit to 1. An overflow should not effect the other status bits. This status should only be cleared by software.

We have defined 7 bits for this status. This is overkill for many units. See below for the bits that are used in the different units.

In general, if the unit has a fixed counter, it will use bit 0. Counter 0 would use the next LSB, and the largest counter would use the MSB.

HA: [4:0] w/ [4] = Counter4 and [0] = Counter 0

IMC: [5:0] w/ [0] = Fixed; [1] = Counter0 and [5] = Counter4

Intel QPI: [4:0] (same as HA)

PCU: [3:0]: [0] = Counter0 and [3] = Counter 3

IO IRP0: [0] = Counter0; [1] = Counter1 IO IRP1: [2] = Counter0; [3] = Counter1

Bus: RootBus0 Device: 5 Function: 6 Offset : F4h			
Bit	Attribute	Default	Description
31:7	RV	0	Reserved
6:0	RW1C	0b	CounterOverflowBitmask This is a bitmask that specifies which counter (or counters) have overflowed. If the unit has a fixed counter, it's corresponding bitmask will be stored at position 0.